INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Diagnosis of asynchronous discrete event systems, a net unfolding approach

Albert Benveniste, Eric Fabre, Claude Jard, and Stefan Haar

N°4181

Mai 2001

THÈME 4

ISSN 0249-6399 ISRN INRIA/RR--4213--FR+ENG

Diagnosis of asynchronous discrete event systems, a net unfolding approach *

Albert Benveniste, Eric Fabre, Claude Jard, and Stefan Haar†

Thème 4 — Simulation et optimisation
de systèmes complexes
Projets Sigma2, Pampa

Rapport de recherche n ° 4181 — Mai 2001 — 40 pages

Abstract: In this paper we formulate asynchronous diagnosis by means of hidden state history reconstruction, from alarm observations. We follow a so-called true concurrency approach, in which no global state and no global time is available. Instead, we use only local states in combination with a partial order model of time, in which local events are ordered if they are either generated on the same site, or related via some causality relation. Our basic mathematical tool is that of *net unfoldings* originating from the Petri net research area. This study was motivated by the problem of event correlation in telecommunications network management.

Key-words: decentralized diagnosis, asynchronous diagnosis, discrete event systems, Petri nets, unfoldings, alarm correlation.

(Résumé : tsvp)

* This work is supported by the RNRT project MAGDA, funded by the Ministère de la Recherche ; other partners of the project are France Telecom R&D, Alcatel, Ilog, and Paris-Nord University.

† IRISA, Campus de Beaulieu, 35042 Rennes cedex, France. A.B., E.F., S.H., are with Inria, C.J. is with CNRS. Corresponding author Albert.Benveniste@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Diagnostic de systèmes à événements discrets asynchrones, une approche par dépliages de réseaux

Résumé : Dans cet article nous étudions le problème du diagnostic pour des systèmes asynchrones. Ce problème est formulé comme un problème de reconstruction de trajactoire d'état à partir des alarmes observées. Nous adoptons un point de vue dit de la "concurrence vraie", ce qui signifie que nous ne manipulons jamais d'états globaux, et que nous utilisons un temps qui a la structure d'un ordre partiel. Notre outil principal est le concept de *dépliage* de réseau de Petri. Nous étudions un certain nombre de variantes de ce problème. Cette étude est motivée par le cas de la corrélation d'alarmes en gestion de réseaux.

Mots-clé : Diagnostic asynchrone, systèmes discrets, réseaux de Petri, dépliage, corrélation d'alarmes.

Contents

1 Introduction

The diagnosis of large, distributed systems is a task of growing importance today. For instance, the increasing complexity and openness of telecommunication networks makes network diagnosis one of the challenging tasks in network management today — network diagnosis is called *event correlation* in this area.

Fault diagnosis in discrete event systems has attracted a significant attention. We refer the reader to [24][10] for an overview of the litterature and introduction to the subject.

Decentralised diagnosis has been identified as a useful approach for the diagnosis of complex systems, in which alarms are sensed within different components of the system. Decentralised diagnosis has been considered by several authors. A thorough study is performed in [10], including both algorithms and their diagnosability properties. In this reference, the notion of a diagnoser is used, meaning that the solution is formulated in terms of a set of communicating machines that have their states labelled by sets of faults, and react to alarm observations and communications. Also, the language oriented framework of Wonham and Ramadge is used [9], and the systems architecture is that of communicating automata, with a "synchronous" communication based on a global time, as revealed by the assumption "A6" therein. Interestingly, the work [10] has been extended by the same authors in [11] toward considering the effect of communication delays in decentralized diagnosis, this is very close to the present paper in which we consider asynchrtonous communications from the very beginning, but adopt a different approach. Issues of communications in diagnosis is also investigated in [25]. Finally, the recent work [26] discussed issues of undecidability for a certain type of decentralized observability, this issue has again some relation with asynchrony. In [4], a different approach is proposed, more in the form of a simulation guided by the observed alarms, again for a model of communicating automata. The solution proposed offers a first attempt to handle the problem of state explosion due to the interleaving of the events involving the different components. Finally, large systems are subject to frequent reconfigurations, updates, and upgrades. Therefore it is desirable and sometimes requested that diagnosis techniques can comply with the situation in which the model used is *dynamically changing* in time. To our knowledge, this is not the case for the existing approaches so far. It is one of the objectives of this paper to investigate this case.

Event correlation in network management is probably the main example of a complex system subject to diagnosis. This area is the subject of a vast litterature, and a number of commercial products are available. We refer the reader to [13] for a survey. There are two main frameworks for most of the methods developed in this area. The first framework relates to rule-based or case-based reasoning, an approach very different from the one we study here. The second one uses a causal model, in which the relation between faulty states and alarm events is modelled in some way or another. The articles [7][8][20] belongs to this family. The authors formulate diagnosis as an optimisation problem, this has some relation with the present approach. The case of event correlation in network management also motivated the series of papers [6][2][3], on which the present paper relies.

In this paper, motivated by the example of telecommunication network diagnosis and management, we focus on diagnosis algorithms that are suitable to truly asynchronous systems. Figure 1 illustrates our purpose by showing two samples of architectures. The first

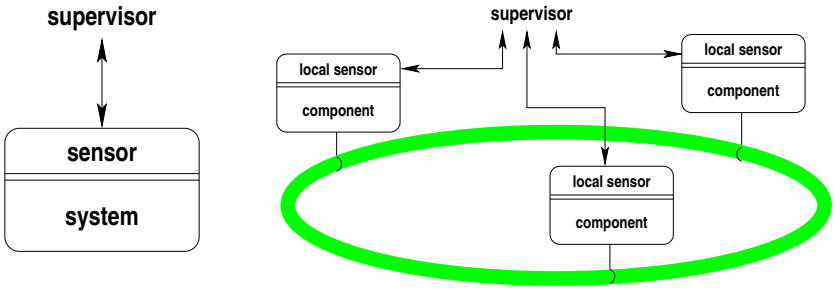

Figure 1: Diagnosis architectures : centralized, and decentralized/asynchronous.

architecture is centralized : one system is globally monitored by one sensor (or, equivalently, one set of synchronized sensors), and one global supervisor performs diagnosis. In the second architecture, the sensoring system is distributed. It has several local sensors, each of them having only a partial view of the overall system, the sensors are not synchronized, alarms are reported to the supervisor asynchronously, and a global supervisor performs diagnosis — this is the typical architecture today in telecommunication systems. The extension to heterogeneous networks, which requires considering distributed supervisors, is discussed in the full paper [1]. In this paper we focus on the second case of figure 1.

We follow an approach based on partial order models of time and making no use of a global notion of state. This approach was introduced in [6][2][3] and is motivated by the figures 2 and 3. Figure 2 depicts a typical fault propagation in a SDH/SONET network management system. The different boxes are network elements, and the different links correspond to the different layers in the SDH/SONET hierarchy. The fault shown is a fault in STM1 port, resulting in the appearance of a symmetric fault at the distant extremity of the link, both faults propagating upward the hierarchy. The architecture shown for the management system has a distributed, asynchronous, set of sensors, but a central supervisor which collects all alarms (there are many of them !). The resulting chronogram of faults and associated emitted alarms is depicted in figure 3, first diagram. The four network elements are shown. In each element, the different lines figure the different layers in the hierarchy. Finally, the observed part of this pattern is shown in figure 3, second diagram. Note that, while events collected on the same sensor are ordered in time, this is not the case for events collected on different sensors. Those events are mutually ordered only when some causality relationship exists, for instance some fault is propagated, or some alarm is broadcast through the network, resulting in a causality relation between its emission and corresponding reception. From the above discussion, the following requirements for our modelling approach emerge, namely :

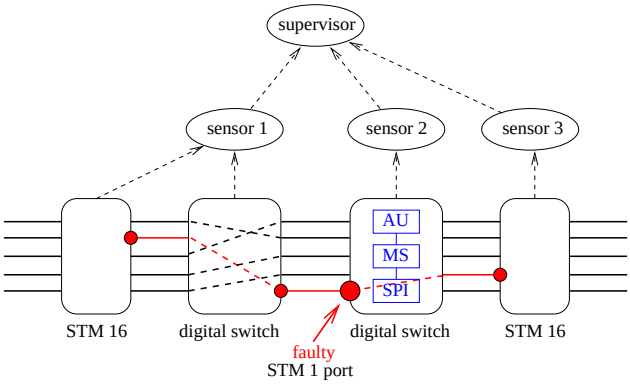

Figure 2: The example of SDH/SONET networks.

- a partial order model of time shall be used, time shall capture 1/ the local time at each local sensor, and possibly, 2/ causal relations between the distributed events ;
- the communication model shall be asynchronous ;
- no global state shall be utilized.

Such an approach was introduced and discussed in details in [3]. But the so-called "Viterbi puzzle" introduced in the latter reference was not completely specified. In particular, the data structure needed to keep track of *all* solutions to the diagnosis problem was not described, only the way *one* solution should be built was discussed.

In this paper we solve the whole problem and provide one possible data structure for the case in which solutions to the diagnosis problem are built *on-line* while alarms are collected (clearly, this is not the only relevant approach, off-line algorithms are also relevant, see [22]). Paper [17] proposes an approach based on synchronizing automata. States for each local automaton are handled in an enumerated way, but the global, product automaton is never built and the local diagnosers work asynchronously in cooperation toward building the overall diagnosis. In this paper we adopt a net unfolding approach [21][15][16]. Net unfoldings were introduced by D. McMillan to allow reachability analyses in an efficient way for Petri nets [21]. They were generalized to other models of concurrency in [16], including synchronized automata (called "synchronous products of transition systems" in the above reference). Net unfoldings are not wellknown in the control community, they have been used for supervisory control in [18][19]. Finally, net unfoldings were recognized a useful structure for proof systems dealing with products of automata with enumerated states [21]. Net unfoldings support useful concepts such as

- *causality,* resulting in partial orders of events ;
- *concurrency,* in order to allow considering local states only, never global states ;

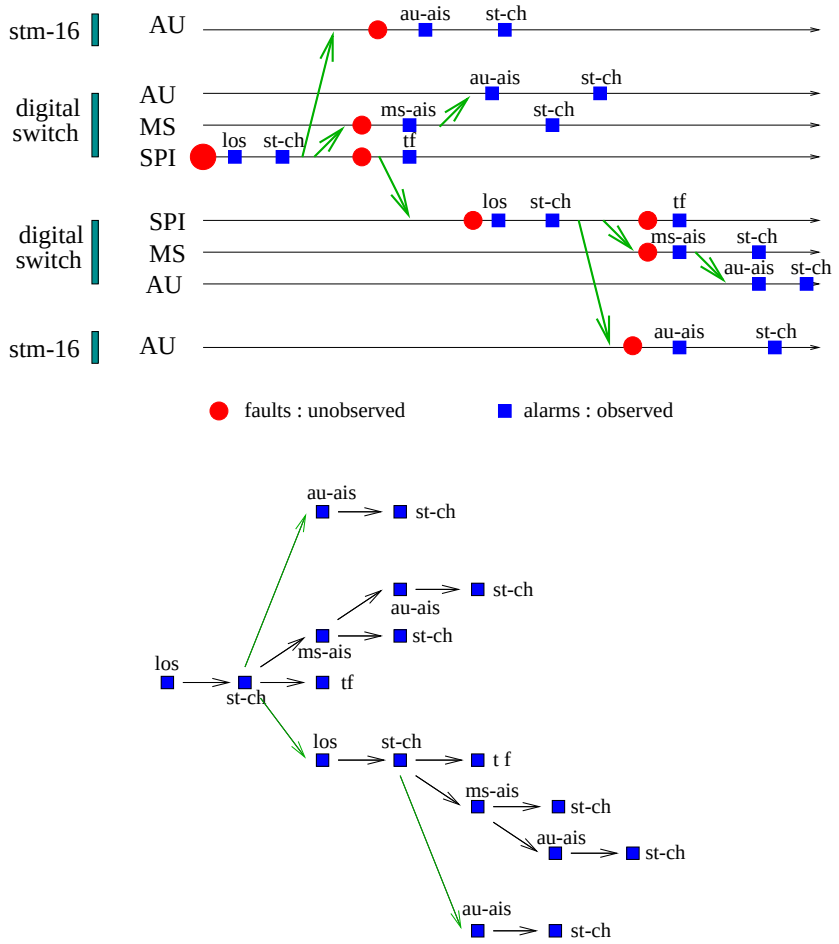

Figure 3: A chronogram of fault and alarm propagation, corresponding observed part.

- *conflict,* to model branching histories, i.e., different histories sharing some prefix; hence conflict relations will allow us to handle the set of all solutions to the diagnosis problem in an efficient way.

When dealing with model based diagnosis, an underlying model is always assumed to be given. Unfortunately, not paying attention to the feasibility of this task may result in a non practical solution. Clearly, getting an appropriate model for the fault and alarm propagation in SDH/SONET telecommunications network is a formidable challenge by itself. The available information typically has the following form :

- Components of the system and their relations are described, in the form of an object oriented model involving classes and their generic inheritance and other relations, and associated diagrams involving deployed instances. No precise behaviour, but only architectures are described at this point.
- Various information regarding components may be available in the form of state transition diagrams showing the behaviour of generic components, at some level of abstraction.
- Information regarding the interaction between components are available in the form of scenarios involving the exchanged messages, and so are typical fault and alarms propagation informations.

Actual instanciation of the classes would then result in some instanciation of the generic components, together with their interconnections, and this would yield the desired model. The important point is that such a construction can be partially automatized, once the classes have been defined. As for describing the behaviour of the different objects, we assume that a *transition relation* is attached to each class as an additional method. This transition relation relates a finite set of previous input and state variables to the corresponding set of current output and state variables of each considered object. State variables of these objects can be either local or shared. Since objects have knowledge of only part of the variables, these transition relations relate *partial* states. Such partial transition relations are called *tiles* in this paper, to refer to puzzle games, a metaphor already used in [3].

The paper is organized as follows. Our basic model of tiles and systems is introduced in section 2, and we lift this model to a Petri net framework. Petri net and their unfoldings are presented in section 3. Diagnosis is discussed in section 4, which constitute the core of this paper. It is formulated as the reconstruction of an unfolding from alarm observations, this unfolding encodes all solutions to our diagnosis problem. Diagnosis is discussed in section 4, using Petri net unfoldings and their variants. In section 5 we introduce a new data structure, called Petri net lattices, and we use them for diagnosis. Lattices are the proper generalization, to Petri net, of the lattice data structures used for the standard Viterbi algorithm for Maximum Likelihood estimation of state trajectories in stochastic automata. Corresponding algorithms are given in section 6. The case of diagnosis with distributed supervisors in discussed in the full paper [1]. Section 7 is devoted to the important case in which the model itself is dynamically changing. Finally we draw some conclusions and perspectives.

2 Tiles and systems

In this section we introduce our mathematical framework. Tiles correspond to partial transitions and systems are defined as a collection of tiles.

2.1 Tiles, systems and their parallel composition

Let $\mathcal{V}$ be a (possibly infinite) set of variables. Each variable $v \in \mathcal{V}$ takes its values in some finite domain $\mathcal{D}_v$. For $V \subseteq \mathcal{V}$, we set $X_V = \prod_{v \in V} \mathcal{D}_v$, and we write $X = X_{\mathcal{V}}$. Elements of X are written x and are called *states* and elements of X_V are denoted by x_V and are called *V-states,* or local states. Note that, for $v \in V$, we have $v(x_V) = v(x) = x_v$. We shall consider local transitions relating local states, very much in the same way transitions relate states in standard automata. These local transitions will be referred to as *tiles* in the sequel.

Formally, a *tile* is a 4-tuple

$$\tau \quad = \quad \langle V, x_V^-, \alpha, x_V \rangle \tag{1}$$

where $V \subset \mathcal{V}$ is a subset of variables, and (x_V^-, α, x_V) is a local transition, relating the previous V-state $x_V^- \in \mathcal{D}_V$ to the current V-state $x_V \in \mathcal{D}_V$, and emitting event α where α ranges over some set A of possible event labels. Due to our particular area of interest, these events will be frequently referred to as *alarms* in the sequel. For τ a tile, we shall sometimes denote by V_τ its set of variables.

A *system* is a triple $\Sigma = \langle V, X_0, \mathcal{T} \rangle$, where $V \subset \mathcal{V}$ is a finite set of variables, X_0 is a set of initial states, and $\mathcal{T}$ is a finite set of tiles and $V = \cup_{\tau \in T} V_\tau$. Systems can be composed to build larger systems. For $\Sigma_i = \langle V_i, X_0^i, \mathcal{T}_i \rangle$, $i = 1, 2$ two systems, their parallel composition $\Sigma_1 \parallel \Sigma_2$ is defined as follows:

$$\Sigma_1 \parallel \Sigma_2 \quad \triangleq \quad \langle V_1 \cup V_2, X_0^1 \cap X_0^2, \mathcal{T}_1 \cup \mathcal{T}_2 \rangle \tag{2}$$

Parallel composition is commutative and associative.

The interleaved sequence of states and alarms

$$x_0, \alpha_1, x_1, \alpha_2, x_2, \ldots, \alpha_k, x_k, \ldots$$

is a *run* of system Σ if $x_0 \in X_0$ and, for each $k > 0$, there exists $\tau = \langle V_\tau, x_{V_\tau}^-, \alpha, x_{V_\tau} \rangle \in \mathcal{T}$ such that,

$$\begin{aligned} \forall v \in V_\tau \quad &: \quad v(x_{k-1}) = v(x_{V_\tau}^-), \alpha_k = \alpha, v(x_k) = v(x_{V_\tau}) \,, \\ \text{and, } \forall v \notin V_\tau \quad &: \quad v(x_{k-1}) = v(x_k) \,. \end{aligned}$$

Since tiles define local transitions, it may be the case that two successive tiles of the considered run, say τ and τ', involve disjoint sets of variables, i.e., modify different local states. In this case, exchanging τ and τ' in the considered run yields another run. This new run is said to be equivalent to the first one, up to an interleaving. The transitive closure of the so defined relation is an equivalence relation. Since we advocate a local view of state and time, *it is advisable not to distinguish runs that are equivalent up to an interleaving.* Partial order semantics is the classical answer to this request, we shall present it in the following section, using a Petri net framework.

This being said, asynchronous diagnosis is formulated as follows: given an alarm sequence $\alpha_1, \alpha_2, \ldots$, reconstruct the set of all runs that have this alarm sequence associated

with them. Decentralized diagnosis with a single supervisor and a distributed set of sensors with asynchronous communications, is the same problem, in which a partially ordered "alarm pattern" is assumed instead of a totally ordered alarm sequence. This is the problem considered from now on.

Finally, distributed diagnosis is formulated as follows. We are given a compound system $\Sigma = \|_{I \in I} \Sigma_i$. Each run of Σ projects onto a local run for each component Σ_i. Each component has its own supervisor, which collects the local alarms of each local run. The different supervisors work concurrently, aiming at constructing, jointly, the set of all runs of Σ which can generate the locally observed alarm sequences. Distributed diagnosis is analyzed in the full paper [1].

2.2 From systems to Petri nets

We shall lift our framework into the more classical framework of Petri nets. This will allow us to use corresponding background to develop our approach. At this stage, we assume the reader familiar with Petri nets. If this fails to be the case, or if the reader feels uncomfortable with some notations, she or he is referred to the next section, where all material we need is exposed. We are given a system $\Sigma = \langle V, X_0, \mathcal{T} \rangle$. We shall represent Σ as a labelled Petri net $\mathcal{N}$ defined as follows:

- The places of $\mathcal{N}$ have the form $p = (v, x_v)$, where v ranges over V, and x_v ranges over the domain of v.
- The transitions of $\mathcal{N}$ are just the tiles $\tau \in \mathcal{T}$. For $\tau = \langle V, x_V^-, \alpha, x_V \rangle$ a tile, we associate to it its alarm component α as a label, and we write $\lambda(\tau) = \alpha$.
- Arrow $(v, x_v) \rightarrow \tau$ belongs to $\mathcal{N}$ iff $1/\ v \in V_\tau$, and $2/\ v(x_{V_\tau}^-) = x_v$. Similarly, arrow $\tau \rightarrow (v', x_{v'})$ belongs to $\mathcal{N}$ iff $1/\ v' \in V_\tau$, and $2/\ v'(x_{V_\tau}) = x_{v'}$.

The so defined net $\mathcal{N}$ is interpreted as a net of capacity one, it is subsequently transformed as usually into a *safe net*[1] by including appropriate auxiliary places and flow relations. This lifting allows us to cast our problem in the framework of Petri nets, which we detail now. The mapping of tiles to transitions is illustrated in figure 4. In this figure, the diagram

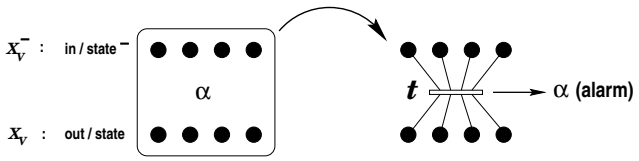

Figure 4: From tiles to transitions.

on the left depicts a tile. Black patches figure valuation of state variables. For V the set

[1] A Petri net is called *safe* if each place can hold at most one token.

of variables of the tile, the patches on the left denote previous state variables $x_v^-, v \in V$, whereas the patches on the right denote current state variables $x_v, v \in V$. On the right hand side, the tile is redrawn as a transition t. The resulting transition fires downward. When firing, it emits an alarm α. See also figure 10.

3 Some background on Petri Nets and their unfoldings

We shall use some basic notions related to Petri Nets and their unfoldings. For the basic definitions on Petri net, we refer the reader to [23][9][12]. Unfoldings are a way to encode the reachable markings of a Petri net by means of a special acyclic Petri net called an occurrence net. A key point is that it is enough to regard this occurrence net as a directed graph in order to compute the reachable markings of the original Petri net. We now collect the notions we need on Petri net and occurrence nets.

3.1 Petri Nets and their products

A *Petri net* consists of a set of *places,* graphically represented by circles and generically denoted by p, a set of *transitions,* graphically represented by bars and generically denoted by t, and a flow relation assigning to each place (transition) a set of input and a set of output transitions (places). The flow relation is graphically represented by arrows leading from places to transitions and from transitions to places. Places and transitions are called *nodes,* generically denoted by n. For n a node, the set of its input and output nodes is denoted by ${}^\bullet n$ and $n^\bullet$, respectively. A place of a net can hold tokens, and a map assigning to each place a number of tokens is called a *marking.* If at a given marking all the input places of a transition hold a token, then the transition can *occur,* which leads to a new marking obtained by removing one token from each input place and adding one token to each output place. In this paper we consider only *safe* Petri nets, in which places can only hold zero or one token. An *occurrence sequence* is a sequence of transitions that can occur in the order specified by the sequence. A Petri net $\mathcal{N} = \{P, P_0, T, \rightarrow\}$ is characterized by its set P of places, its set T of transitions, its flow relation $\rightarrow \subset (P \times T) \cup (T \times P)$, and the subset $P_0 \subseteq P$ which composes the *initial marking* of the net. A *labelling* of Petri net $\mathcal{N}$ is a map $\lambda : T \longmapsto A$, where A is some finite alphabet. A Petri net equipped with a labelling is called a labelled Petri net.

Definition 1 (product of Petri nets) *The* product $\mathcal{N}_1 \times \mathcal{N}_2$ *of the two labelled Petri nets* $\mathcal{N}_i = \{P_i, P_{0,i}, T_i, \rightarrow_i, \lambda_i\}$, $i = 1, 2$, *is a labelled Petri net defined as follows :*

$$\begin{aligned} \mathcal{N}_1 \times \mathcal{N}_2 \quad \triangleq \quad & \{P, P_0, T, \rightarrow, \lambda\} \text{ , where } P = P_1 \cup P_2 \text{, } P_0 = P_{0,1} \cup P_{0,2} \text{ ,} \\ & \text{and } t \in T \text{ iff :} \end{aligned} \tag{3}$$

case (i) : $\exists t_i \in T_i$ *for some* $i = 1, 2$, *such that*

$$\begin{aligned} \lambda(t) = \lambda_i(t_i) \quad & \in \quad A_i \setminus A_j \text{ , } j \neq i \text{ , and} \\ {}^\bullet t = {}^\bullet t_i \quad & , \quad t^\bullet = t_i^\bullet \end{aligned}$$

case (ii) : *for each* $i = 1, 2$, $\exists t_i \in T_i$ *such that*

$$\lambda(t) = \lambda_1(t_1) = \lambda_2(t_2) \text{ , and}$$
$${}^\bullet t = {}^\bullet t_1 \cup {}^\bullet t_2 \quad , \quad t^\bullet = t_1^\bullet \cup t_2^\bullet$$

The product is associative and commutative. In case (i) only one net fires a transition and this transition has a private label, while the two Petri nets synchronise on transitions with identical labels in case (ii). Note that the two Petri nets can have shared places, this deviates from the usual notion of product [16][12].

A link with the parallel composition of systems. Let us establish a link between this notion of a product of Petri nets, and our previous definition (2) for the parallel composition of systems. Consider two systems Σ_1 and Σ_2, and assume they share variables ($V_1 \cap V_2 \neq \emptyset$) but they have distinct sets of alarms, this is the case of interest for practical distributed diagnosis applications. Then corresponding Petri nets $\mathcal{N}_1$ and $\mathcal{N}_2$ have shared places, but distinct labels for their events, and therefore, when considering their product $\mathcal{N}_1 \times \mathcal{N}_2$, only case (i) applies. As a consequence, in this case, *product net* $\mathcal{N}_1 \times \mathcal{N}_2$ *is the net associated with the composed system* $\Sigma_1 \| \Sigma_2$. Note that in our approach, interaction between subsystems occur through shared places, unlike in DES theory, where only events and their synchronization is considered for defining products of subsystems.

A link with the diagnosis problem. Consider the case in which net $\mathcal{N}_2$ has its event label set contained in that of net $\mathcal{N}_1$, meaning that events produced by $\mathcal{N}_2$ "could have been" produced by $\mathcal{N}_1$. On the other hand, we assume that $\mathcal{N}_1$ and $\mathcal{N}_2$ have distinct places, say, places of $\mathcal{N}_2$ are dummy and have no particular concrete signification. When computing the product $\mathcal{N}_1 \times \mathcal{N}_2$, then only case (ii) of synchronizing events applies, and this product will in particular compute those behaviours of $\mathcal{N}_1$ which can "explain" the observed behaviours of $\mathcal{N}_2$. This is very similar to the diagnosis problem we shall investigate later.

3.2 Representing the runs of a Petri net via unfoldings

In this subsection we consider the problem of representing all the runs of a Petri net, using the notion of occurrence net and unfolding.

Occurrence nets – definition, terminology, and notations

Given two nodes n and n' (place or transition) of a Petri net, we say that n *causes* n', written $n \preceq n'$, if either $n' = n$ or there is a path of arrows from n to n'. We say that n and n' are in *conflict,* written $n \# n'$, if there is a place m, different from n and n', from which one can reach n and n', exiting m by different arrows. Finally we say that n and n' are *concurrent,* written $n \perp\!\!\!\perp n'$, if neither $n \preceq n'$, nor $n \preceq n'$, nor $n \# n'$ hold. For n a node, we write $\perp\!\!\!\perp(n)$ to denote the set of nodes that are concurrent with n. An *occurrence net* is a Petri net satisfying the following properties :

1. the net, seen as a directed graph, has no circuit ;
2. every place has at most one input transition ;
3. no node is in self-conflict, i.e., $n\#n$ does not hold.

Occurrence nets can be infinite. We restrict ourselves to those in which every event has at least one input place, and in which the arrows cannot be followed backward infinitely from any point (this is referred to as *well-foundedness*). It follows that, by following the arrows backward we eventually reach a place without predecessors, these are the *minimal places* of the occurrence net.

To distinguish occurrence nets from other Petri nets, we shall use specific terminology and notations, mostly borrowed from [15]. Occurrence nets are generically denoted by $\mathcal{U}$, their places are called *conditions,* they are generically denoted by b, the set of conditions is denoted by B, and the set of minimal conditions of $\mathcal{B}$ is denoted by $\min(B)$. Also, their transitions are called *events,* they are generically denoted by e, and the set of events is denoted by E.

A *cut* of an occurrence net is a set of conditions $\mathbf{c}$ satisfying the following two properties : $\mathbf{c}$ is a *co-set* (any two elements of $\mathbf{c}$ are concurrent), and $\mathbf{c}$ is maximal (it is not properly included in any other co-set). A *configuration* is a set of nodes κ satisfying the following two properties : κ is causally closed (if $n \in \kappa$ and $n' \prec n$, then $n' \in \kappa$), conflict-free (no two nodes of κ are in conflict), and, when seen as a set of nodes, a configuration is a union of cuts. Furthermore we require for convenience that all maximal nodes (if any) of configurations shall be conditions. Figure 5 shows an example of a configuration, figure 6 depicts different

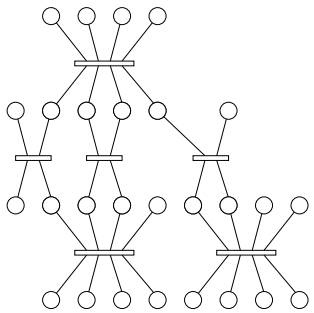

Figure 5: A configuration.

cuts for the configuration of figure 5 (the cuts are composed of the black patches), and figure 7 shows an example of an occurrence net. The sources of the conflict is depicted in black. Two different configurations branch from this conflict, the grey one and the white one. The conflict is indicated by the # symbol.

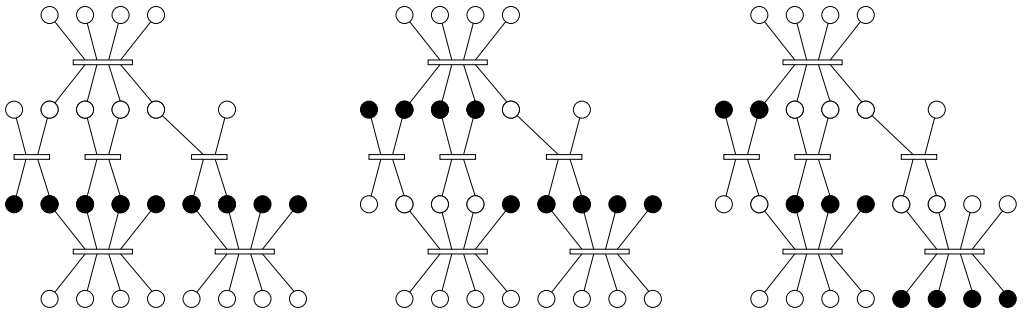

Figure 6: Showing three different cuts for the configuration of figure 5.

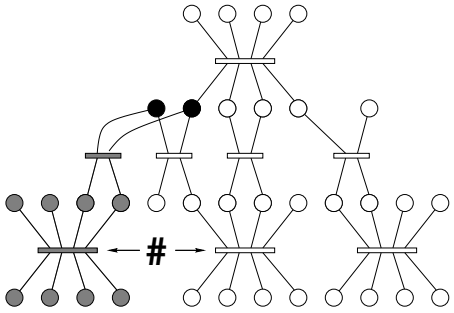

Figure 7: Occurrence net.

Unfoldings

Given a Petri net $\mathcal{N} = (P, P_0, T, \rightarrow)$, we associate to it a set of labeled occurrence nets, called the *branching processes* of $\mathcal{N}$. The conditions of these nets are labeled with places of $\mathcal{N}$, and their events are labeled with transitions of $\mathcal{N}$. An *homomorphism* from Petri net $\mathcal{N}'$ to Petri net $\mathcal{N}''$ is a map $\varphi : P' \cup T' \mapsto P'' \cup T''$, which preserves the type of node (place or transitions) and flow relation, i.e., which satisfies $\varphi(P') \subseteq P''$, $\varphi(T') \subseteq T''$, $p' \rightarrow' t' \Rightarrow \varphi(p') \rightarrow'' \varphi(t')$, and $t' \rightarrow' p' \Rightarrow \varphi(t') \rightarrow'' \varphi(p')$, where $\rightarrow', \rightarrow''$ denote the flow relations for $\mathcal{N}', \mathcal{N}''$, respectively.

Definition 2 (branching process) *A* branching process *of a Petri net $\mathcal{N} = (P, P_0, T, \rightarrow)$ is a labelled occurrence net $\mathcal{B} = (B, E, \rightarrow, \varphi)$, where the labelling function φ, with domain $B \cup E$, satisfies the following properties:*

(i) φ is an homomorphism from $\mathcal{B}$ to $\mathcal{N}$.

(ii) The restriction of φ to $min(B)$, seen as a directed graph, is a bijection between $min(B)$ and P_0.

(iii) $\forall e_1, e_2 \in E$, ${}^\bullet e_1 = {}^\bullet e_2$ *and* $\varphi(e_1) = \varphi(e_2)$ *together imply* $e_1 = e_2$.

Condition *(iii)* expresses that the unfolding of the flow relation is performed without duplication of events. The set of all branching processes is uniquely defined, up to an isomorphism (i.e., a renaming of the conditions and events), and we shall not distinguish isomorphic branching processes. Branching processes are partially ordered as follows :

$$\mathcal{B} \preceq \mathcal{B}' \quad \text{iff} \quad \text{there exists an injective homomorphism from } \mathcal{B} \text{ into } \mathcal{B}' , \tag{4}$$

note that this homorphism preserves the labelling functions. By theorem 23 of [14], there exists (up to an isomorphism) a unique maximum branching process according to $\preceq$,

$$\text{we call it the } \textbf{\textit{unfolding}} \text{ of } \mathcal{N}, \text{ and denote it by } \mathcal{U}_{\mathcal{N}} . \tag{5}$$

Fixing $\mathcal{U}_{\mathcal{N}}$ provides a universal naming of the nodes of all branching processes of $\mathcal{N}$. Having this naming fixed,

$$\mathcal{B} \preceq \mathcal{B}' \quad \text{reduces to} \quad \mathcal{B} \subseteq \mathcal{B}' ,$$

for $\mathcal{B}, \mathcal{B}' \subset \mathcal{U}_{\mathcal{N}}$. Therefore we identify partial order (4) with inclusion in the sequel. Configurations of $\mathcal{U}_{\mathcal{N}}$ model partial runs of $\mathcal{N}$, cuts correspond to states, and co-sets model local states, see figure 6. Hence $\mathcal{U}_{\mathcal{N}}$ adequately represents all runs of $\mathcal{N}$. Also, it is easily checked that runs for system Σ defined with tiles as in section 2 are mapped to maximal configurations of $\mathcal{U}_{\mathcal{N}}$, for the Petri net $\mathcal{N}$ obtained from Σ by the technique of subsection 2.2.

4 Diagnosis in the framework of Petri Nets

We are given the following objects.

- A labelled Petri net $\mathcal{N} = \{P, P_0, T, \rightarrow, \lambda\}$, where the range of the labelling map λ is the set of possible *alarms,* denoted by A, and
- its unfolding $\mathcal{U}_{\mathcal{N}} = \{B, E, \rightarrow, \varphi\}$, where the different objects have been introduced in subsection 3.2.

Note the following chain of labelling maps :

$$\underbrace{E}_{\text{events}} \xrightarrow{\varphi} \underbrace{T}_{\text{transitions}} \xrightarrow{\lambda} \underbrace{A}_{\text{alarms}} \quad : \quad e \longmapsto \varphi(e) \longmapsto \lambda(\varphi(e)) \stackrel{\Delta}{=} \Lambda_\alpha(e) , \tag{6}$$

which defines the *alarm label* of event e (or "alarm", for short, when no confusion can occur), we denote it by $\Lambda_\alpha(e)$.

We assume some run of net $\mathcal{N}$, i.e., some configuration κ of its unfolding $\mathcal{U}$. And we assume that we only observe the *alarm events* of configuration κ, meaning that, for event

$e \in \kappa$, we only observe $\Lambda_\alpha(e)$. Our task consists in reconstructing all configurations of the unfolding of the considered net, which could give raise to the above mentioned observations. We rephraze this by saying that the desired configurations "explain these observations", and we interpret this as a *diagnosis problem.* To gain an intuition of this, just imagine that we try to recover the diagram on the top of figure 3, from observing the diagram on the bottom of the same figure.

As said before, only alarms are observed, but we also need to specify which information regarding *causalities* between the different alarm events our observation system can capture :

(a) For the simplest design, we assume that we have a single sensor observing sequentially some interleaving $\alpha_1, \alpha_2, \ldots$ ($\alpha_i \in A$), of the alarm events belonging to the considered configuration. We assume that it is not the case that $i' < i$ and α_i caused $\alpha_{i'}$, i.e., the observed interleaving is consistent with the causalities specified by the underlying Petri net.

(b) For a more sophisticated design, we assume that we have a finite set of sensors labelled by $j \in J$. These sensors work concurrently and independently, and sensor j observes sequentially some interleaving $\alpha_1^j, \alpha_2^j, \ldots$ ($\alpha_i^j \in A$), of the alarm events relevant to its site. Again, we assume that, locally to each site, the observed interleaving is consistent with the underlying causalities.

(c) More generally, we assume that we have some sensoring system able to observe alarm events, partially ordered in a way which does not contradict the way they were causally produced.

Case (a) is typical of a diagnosis using a centralized supervisor, in which alarm events are collected in sequence by respecting causalities — a typical architecture in network management systems today. Case (b) is an extension of (a) to diagnosis via distributed supervision — the preferred architecture for future network management systems in which heterogeneous network management systems will cooperate. Finally, case (c) subsumes case (b) to its important assumptions for our purpose. All this is formalized next.

4.1 Problem statement

To formalize the above discussed partial capture of causalities, we need the notion of *extension* of an occurrence net.

Definition 3 (extension of an occurrence net) *Given a labelled occurrence net, we call an* extension *of it any labelled occurrence net obtained by adding, to this net, conditions and flow relations but not events.*

An occurrence net induces a labelled partial order on the set of its events. Extending this occurrence net according to definition 3 induces an extension of this labelled partial order [2].

[2] Recall that the labelled partial order $(X, \preceq)$ is an *extension* of labelled partial order $(X', \preceq')$ if labelled sets X and X' are isomorphic, and $\preceq \supseteq \preceq'$ holds. When $(X, \preceq)$ is a total order, we call it a *linear* extension of $(X', \preceq')$.

Then, we need to formalize what we mean by "observing alarms only". To understand why the following framework is required, the reader is referred to (6) and the related explanations. We consider *alarm labelled* occurrence nets of the form $\mathcal{U} = \{B, E, \rightarrow, \lambda\}$, in which the labelling map $\Lambda_\alpha(e), e \in E$, takes its values in the set A of possible alarms.

Definition 4 (alarm-isomorphic occurrence nets) *Two alarm labelled occurrence nets* $\mathcal{U} = \{B, E, \rightarrow, \Lambda_\alpha\}$ *and* $\mathcal{U}' = \{B', E', \rightarrow', \Lambda'_\alpha\}$ *are called* alarm-isomorphic *if there exists an isomorphism* ψ*, from* $\{B, E, \rightarrow\}$ *onto* $\{B', E', \rightarrow'\}$ *seen as directed graphs, which preserves the alarm labels, i.e., such that* $\forall e \in E : \Lambda'_\alpha(\psi(e)) = \Lambda_\alpha(e)$.

Two alarm-isomorphic occurrence nets can be regarded as identical if we take into account the alarm labels of their events, but ignore their conditions.

Definition 5 (alarm pattern) *Consider a labelled net* $\mathcal{N}$ *and its unfolding* $\mathcal{U}_\mathcal{N}$*. A labelled occurrence net* $\mathcal{A}$ *is called an* alarm pattern of $\mathcal{N}$ *if:*

1. *Its labelling map takes its value in alphabet* A *of alarms,*
2. $\mathcal{A}$ *is itself a configuration (it is conflict free), its set of conditions is disjoint from that of* $\mathcal{U}_\mathcal{N}$*, and*
3. *There exists a configuration* κ *of* $\mathcal{U}_\mathcal{N}$ *such that* $\mathcal{A}$ *and* κ *possess extensions that are alarm-isomorphic.*

Assuming for $\mathcal{A}$ a set of places disjoint from that of $\mathcal{U}_\mathcal{N}$ aims at reflecting that alarm patterns vehicle no information regarding hidden states of the original net. This justifies condition 2. Keeping in mind definitions 3 and 4 and the cases (a,b,c) of the discussion before, condition 3 expresses that κ can explain $\mathcal{A}$. For instance: in case (a), $\mathcal{A}$ itself is an extension of κ; in case (b) each local part of $\mathcal{A}$ is an extension of κ, but causalities relating alarm events received by different sensors are lost; case (c) expresses that κ and $\mathcal{A}$ possess compatible partial orders.

For $\mathcal{A}$ a given alarm pattern of $\mathcal{N}$, we denote by

$$\mathsf{diagnosis}\,(\mathcal{A}) \tag{7}$$

the set of configurations κ of $\mathcal{U}_\mathcal{N}$, satisfying the conditions 1,2,3 of definition 5. In the next subsection, we propose an adequate data structure to represent the set $\mathsf{diagnosis}\,(\mathcal{A})$, we call it a *diagnosis net.*

4.2 Diagnosis nets

In occurrence nets, configurations are causally closed and conflict free unions of cuts. Hence configurations are conveniently characterized using causality and conflict relations. Therefore, a first natural idea is to represent $\mathsf{diagnosis}\,(\mathcal{A})$ by

$$\begin{array}{c}\text{the minimal subnet of unfolding } \mathcal{U}_\mathcal{N}\text{, containing}\\ \text{all configurations} \in \mathsf{diagnosis}\,(\mathcal{A})\text{; we denote it by } \mathcal{U}_\mathcal{N}(\mathcal{A}).\end{array} \tag{8}$$

Subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ inherits canonically by restriction, of the causality, conflict, and concurrence relations defined on $\mathcal{U}_{\mathcal{N}}$. Net $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ contains all configurations belonging to $\mathsf{diagnosis}\,(\mathcal{A})$, but unfortunately it also contains undesirable maximal configurations *not* belonging to $\mathsf{diagnosis}\,(\mathcal{A})$, as the figure 8 shows. In this figure, we show an unfolding $\mathcal{U}$ on the left

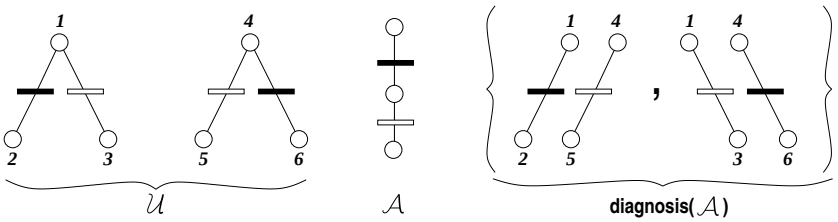

Figure 8: *Showing* $\mathcal{U}, \mathcal{A}$, *and* $\mathsf{diagnosis}\,(\mathcal{A})$.

hand side. In the middle, we show a possible associated alarm pattern $\mathcal{A}$. Alarm labels are figured by colors (black and white). The set $\mathsf{diagnosis}\,(\mathcal{A})$ is shown on the right hand side, it comprises two configurations. Unfortunately the minimal subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ of the original unfolding $\mathcal{U}$ which contains $\mathsf{diagnosis}\,(\mathcal{A})$, is indeed identical to $\mathcal{U}$! Undesirable configurations are $\{(1, t_{12}, 2), (4, t_{46}, 6)\}$ and $\{(1, t_{13}, 3), (4, t_{45}, 5)\}$ (in these statements, t_{12} denotes the transition separating states 1 and 2). But configuration $\{(1, t_{12}, 2), (4, t_{46}, 6)\}$ is such that its two transitions t_{12}, t_{46} explain the same alarm event in $\mathcal{A}$, and the same holds for the other undesirable configuration. The idea is to turn the relation "explain the same alarm event" into a conflict relation, since this would prevent $\{(1, t_{12}, 2), (4, t_{46}, 6)\}$ and $\{(1, t_{13}, 3), (4, t_{45}, 5)\}$ from being configurations in this case! Let us formalize this idea.

Referring to the definition 5 of alarm patterns, for every $\kappa \in \mathsf{diagnosis}\,(\mathcal{A})$, denote by ψ_κ the alarm-isomorphism mapping the events of κ onto those of $\mathcal{A}$. For a an event of an alarm pattern $\mathcal{A}$, we denote by $\mathsf{diagnosis}\,(a)$ the set $\{\psi_\kappa^{-1}(a) \,:\, \kappa \in \mathsf{diagnosis}\,(\mathcal{A})\}$, it is the set of events belonging to $\mathsf{diagnosis}\,(\mathcal{A})$ which explain a. By abuse of notation, we also denote by $\mathsf{diagnosis}\,(a)$ the corresponding set of events in subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$. We define the following $\mathcal{A}$-conflict relation on subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ as follows, we write it $\#_{\mathcal{A}}$:

Definition 6 ($\mathcal{A}$-conflict relation) *Relation $\#_{\mathcal{A}}$ is the weakest relation satisfying the following three conditions :*

1. *If $p \# q$, then $p \#_{\mathcal{A}} q$, i.e., $\#_{\mathcal{A}}$ is stronger than $\#$;*
2. *If $\exists a \in \mathcal{A}$ such that $t \in \mathsf{diagnosis}\,(a)$ and $t' \in \mathsf{diagnosis}\,(a)$, then $t \#_{\mathcal{A}} t'$, i.e., sharing an alarm event is a source of conflict ;*
3. *If $p \#_{\mathcal{A}} q$ and $p \preceq p'$, $q \preceq q'$, then $p' \#_{\mathcal{A}} q'$, i.e., $\#_{\mathcal{A}}$ is causally closed.*

We equip the subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ of nodes with the causality relation $\preceq$ inherited from the original unfolding $\mathcal{U}_{\mathcal{N}}$, together with the above defined reinforced conflict relation $\#_{\mathcal{A}}$. Finally, we write $p \perp\!\!\!\perp_{\mathcal{A}} q$ if neither $p \preceq q$, nor $q \preceq p$, nor $p \#_{\mathcal{A}} q$ holds. The following theorem holds :

Theorem 1 *Let $\mathcal{U}_{\mathcal{N}}$ be the unfolding of some Petri net $\mathcal{N}$, $\mathcal{A}$ an associated alarm pattern, and let* diagnosis $(\mathcal{A})$ *be defined as in (7). Equip $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ with the reinforced conflict relation $\#_{\mathcal{A}}$ introduced in definition 6. Then the triple $\mathcal{U}_{\mathcal{N},\mathcal{A}} = \{\mathcal{U}_{\mathcal{N}}(\mathcal{A}), \preceq, \#_{\mathcal{A}}\}$ is an adequate representation of* diagnosis $(\mathcal{A})$*, as it possesses exactly* diagnosis $(\mathcal{A})$ *as its set of maximal configurations, we write this*

$$\text{diagnosis}\,(\mathcal{A}) \quad \sim \quad \mathcal{U}_{\mathcal{N},\mathcal{A}}$$

Proof: it is organized into several steps.

1. *Every $\kappa \in$* diagnosis $(\mathcal{A})$ *is a configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$.*

   By definition of diagnosis $(\mathcal{A})$, no two nodes of κ explain the same alarm event of $\mathcal{A}$. On the other hand, κ, being a configuration of $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ equipped with its original conflict relation $\#$, is causally closed. Hence no two nodes of κ are causally related to nodes explaining the same alarm event $a \in \mathcal{A}$. Since κ is already a configuration of $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ equipped with its original conflict relation $\#$, conditions 2 and 3 of definition 6 imply that κ is a configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$.

2. Every $\kappa \in$ diagnosis $(\mathcal{A})$ is a *maximal* configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$.

   Assume this is not true, then there exists some $\kappa \in$ diagnosis $(\mathcal{A})$ which is not a maximal configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$. Hence $\kappa \subset \kappa', \kappa \neq \kappa'$, for some maximal configuration $\kappa' \in \mathcal{U}_{\mathcal{N},\mathcal{A}}$. Then two cases can occur. Either the suffix $\kappa' \setminus \kappa$ is not a subset of any element of diagnosis $(\mathcal{A})$, hence it can be removed from $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$, therefore contradicting the minimality of $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$. Or some node of $\kappa' \setminus \kappa$ belongs to some element of diagnosis $(\mathcal{A})$, but then it explains some alarm event of $\mathcal{A}$, hence it must be in conflict with the node of κ explaining the same alarm event of $\mathcal{A}$, this contradicts the fact that κ' is a configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, equipped with the reinforced conflict relation $\#_{\mathcal{A}}$.

3. *Every maximal configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ is an element of* diagnosis $(\mathcal{A})$.

   Assume there exists some maximal configuration κ of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ which does not belong to diagnosis $(\mathcal{A})$. We have already seen that κ cannot contain an element of diagnosis $(\mathcal{A})$. Then the following cases can occur. Either κ has some suffix which has empty intersection with every element of diagnosis $(\mathcal{A})$, hence it can be removed from $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$, therefore contradicting again the minimality of $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$. Or κ is a union of subconfigurations of elements of diagnosis $(\mathcal{A})$. Then either two different nodes of κ explain the same alarm event, and this prevents κ from being a configuration with respect to the enhanced conflict relation $\#_{\mathcal{A}}$, or no such two nodes exist, and then κ must be a strict subconfiguration of some element of diagnosis $(\mathcal{A})$, hence it is not a maximal configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$. ◇

The structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ introduced in theorem 1 is not a proper occurrence net, as its conflict relation is not inherited from the topology of the underlying bipartite graph. However the restriction of this structure to its set of events is an *event structure* according to Winskel's definition [27].

However, structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ lacks algebraic properties, hence it is more difficult reasoning with it. Therefore we shall introduce a less compact, but more elegant representation of $\mathsf{diagnosis}\,(\mathcal{A})$, amenable of algebraic manipulations that will be instrumental in handling the distributed case.

Theorem 2 *Let $\mathcal{N}$, $\mathcal{U}_{\mathcal{N}}$, $\mathcal{A}$, and $\mathsf{diagnosis}\,(\mathcal{A})$ be as in theorem 1, and consider the unfolding $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$. Then erasing, in the set of all configurations of $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, the conditions labelled with nodes from $\mathcal{A}$, yields the set $\cup_{\mathcal{A}'\preceq\mathcal{A}}\,\mathsf{diagnosis}\,(\mathcal{A}')$, where $\mathcal{A}'$ ranges over the set of the prefixes of $\mathcal{A}$. Hence $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$ is an adequate representation of $\cup_{\mathcal{A}'}\,\mathsf{diagnosis}\,(\mathcal{A}')$, written*

$$\bigcup_{\mathcal{A}'\preceq\mathcal{A}} \mathsf{diagnosis}\,(\mathcal{A}') \quad\sim\quad \mathcal{U}_{\mathcal{N}\times\mathcal{A}}\,.$$

This theorem is illustrated in the figure 9, which continues our running example. The reader

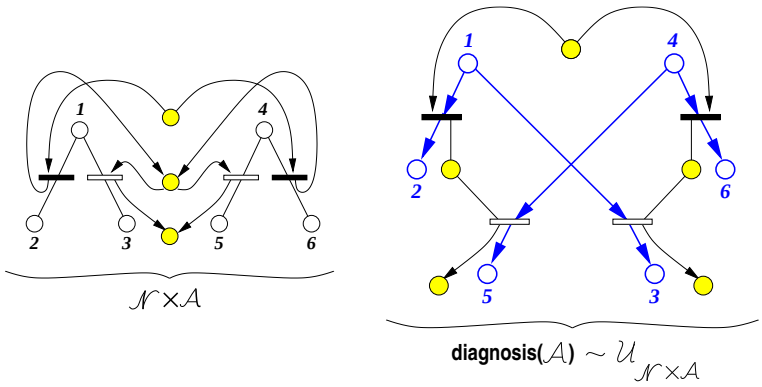

Figure 9: *representing* $\mathsf{diagnosis}\,(\mathcal{A})$ *via* $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$.

should compare this figure with figure 8, and note that the restriction, to the nodes of $\mathcal{N}$, of unfolding $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, has exactly $\mathsf{diagnosis}\,(\mathcal{A})$ as its set of configurations.

Proof of theorem 2 : it relies on the following claims.

1. *Let κ be a configuration of $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$; then erasing, in κ, the conditions labelled by nodes from $\mathcal{A}$, yields an element of $\mathsf{diagnosis}\,(\mathcal{A}')$, where $\mathcal{A}'$ is obtained by erasing, in κ, the conditions labelled by nodes from $\mathcal{N}$.*

   To prove this claim, we use in detail the definition 1 of the product of Petri nets, and the definition 2 of branching processes. Since $\mathcal{N}$ and $\mathcal{A}$ possess the same labelling alphabet A, only case (ii) of definition 1 applies, i.e., both $\mathcal{N}$ and $\mathcal{A}$ must synchronize at each transition of their product $\mathcal{N}\times\mathcal{A}$. Using this remark and conditions (iii) and (ii) of definition 2, erasing, in κ, the conditions labelled by places from $\mathcal{N}$, yields a

configuration of $\mathcal{U}_{\mathcal{A}} = \mathcal{A}$ (we use the obvious fact that the unfolding of an occurrence net is this net itself). But the configurations of $\mathcal{A}$ are just its prefixes, hence the so obtained configuration is some prefix, $\mathcal{A}'$, of $\mathcal{A}$. Using again the fact that all transitions of $\mathcal{N} \times \mathcal{A}$ are synchronizing transitions, erasing, in κ, the conditions labelled by conditions from $\mathcal{A}$, yields a configuration of $\mathcal{U}_{\mathcal{N}}$. By definition 5 of alarm patterns, this configuration is an element of $\mathsf{diagnosis}(\mathcal{A}')$.

2. *For $\mathcal{A}'$ an arbitrary prefix of $\mathcal{A}$, any element of* $\mathsf{diagnosis}(\mathcal{A}')$ *is obtained by erasing, in some configuration κ of $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, the conditions that are labelled by nodes from $\mathcal{A}'$.*

   It is enough to prove the above claim for the special case $\mathcal{A}' = \mathcal{A}$. By theorem 1, we know that $\mathsf{diagnosis}(\mathcal{A})$ coincides with the set of maximal configurations of the structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$. From the definition 6 of enhanced conflict relation $\#_{\mathcal{A}}$, we immediately see that any maximal configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ is obtained by erasing, in some configuration κ of $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, the conditions that are labelled by nodes from $\mathcal{A}'$. This proves the claim, and finishes the proof of the theorem. ◇

The data structures introduced in theorems 1,2 are generically called ***diagnosis nets*** in the sequel.

The above techniques are illustrated on the example of figure 10. The example consists of two interacting components, specified in two separate boxes. We show the tiles $\tau_{i,j}$, where $i = 1, 2$ denotes the component. Self-repair can occur (shown by ρ), as well as failure of component 1 to provide its service to component 2. Hence the two components interact via their shared variable $\mathbf{service}_{12}$. Below we show the Petri net translation. The correspondence between places and pre/post-conditions is indicated in the tile-based specification, for instance the statement "state_1 = enabled (1)" means that place (1) of the Petri net translation corresponds to post-condition "state_1 = enabled" in the tile-based specification. Place 7 does not appear in this correspondence, but we have added the complementary branch $\{\alpha \rightarrow \ 7 \ \rightarrow \beta\}$ to make the Petri net safe. The initial marking is composed of the places filled in grey.

We show in figure 11 an illustration of the above approach for diagnosis. In this figure we show the Petri net $\mathcal{N}$ of figure 10 (left), a prefix of its unfolding $\mathcal{U}_{\mathcal{N}}$ (middle), and an associated alarm pattern $\mathcal{A}$ (right). The subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ is composed of the conditions and events in grey, it contains in particular $\rho \leftarrow \beta \rightarrow \alpha \rightarrow \rho$ as a maximal configuration, but this configuration does not belong to $\mathsf{diagnosis}(\mathcal{A})$, since ρ appears twice. We show in dashed-thick the additional conflict relation between the two events labelled by the same alarm ρ, resulting in the event structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ of theorem 1. It is easily checked that $\mathsf{diagnosis}(\mathcal{A})$ is adequately represented by $\mathcal{U}_{\mathcal{N},\mathcal{A}}$.

5 Petri net lattices, and their use for asynchronous diagnosis

Unfoldings can be considered and used for automata as well. They are branching structures in which the shared prefixes for different runs of the automaton are represented only once.

$\tau_{1,1}$	pre-condition :	state_1 = enabled (1)
	message :	alarm(β)
	post-condition :	state_1 = disabled (2) $\wedge$ $\mathbf{service}_{12}$ = disabled (3)
$\tau_{1,2}$	pre-condition :	state_1 = disabled (2)
	message :	self-repair(ρ)
	post-condition :	state_1 = enabled (1)
$\tau_{2,1}$	pre-condition :	$\mathbf{service}_{12}$ = disabled (3) $\wedge$ service_2 = enabled (4)
	message :	alarm(α)
	post-condition :	service_2 = disabled (5)
$\tau_{2,2}$	pre-condition :	service_2 = enabled (4)
	message :	alarm(α)
	post-condition :	state_2 = disabled (6)
$\tau_{2,3}$	pre-condition :	service_2 = disabled (5)
	message :	self-repair(ρ)
	post-condition :	service_2 = enabled (4)

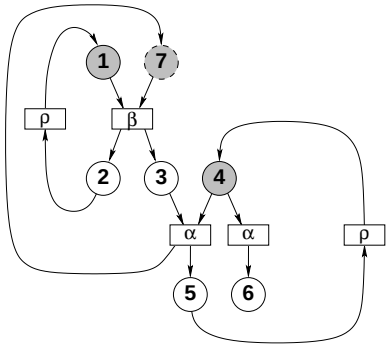

Figure 10: Example 1: description using tiles, and its Petri net translation.

But the reconstruction of hidden runs from observations, for automata, typically use a different structure, not unfoldings. They use the *lattice* representation for trajectories, as illustrated in figure 12. This diagram shows a Petri net wich is in fact an automaton (left). Its unfolding is shown in the middle. We can safely merge the two branches {c $\rightarrow \beta \rightarrow$ a $\rightarrow \alpha \rightarrow$c} and {c $\rightarrow \beta \rightarrow$ b $\rightarrow \alpha \rightarrow$c} since

(ℓ_1) their maximal condition is labelled by the same state (namely c), hence they possess identical continuations, and

(ℓ_2) they explain the same alarm sequence.

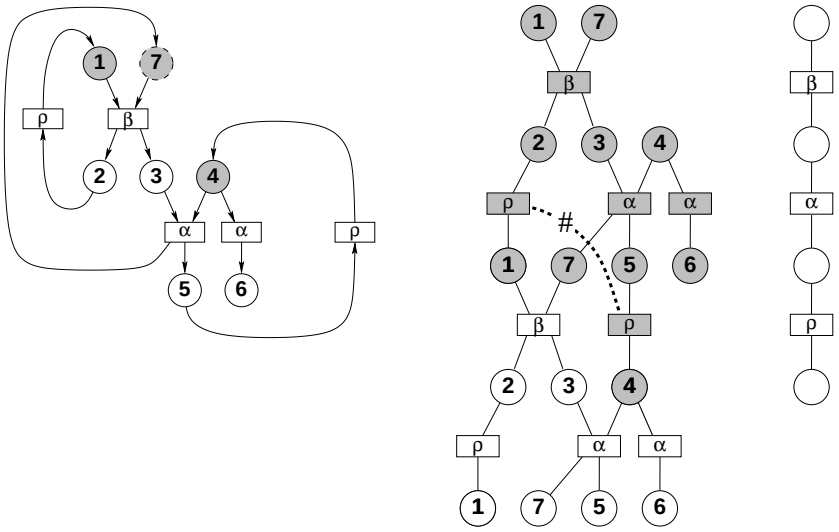

Figure 11: Example 1, unfolding and diagnosis, an illustration of theorem 1.

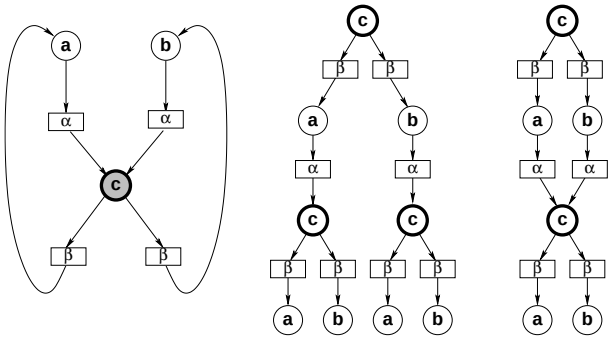

Figure 12: *Automata: unfoldings and lattices.*

The resulting Petri net on the right is called a *lattice.* Lattices are not occurrence nets any more, they are rather infinite Petri nets having both branching and joining conditions, but no circuit. Corresponding structures for general safe Petri nets are not known, at least to our knowledge. We shall devote this section to their construction, and their use for diagnosis.

We build upon the work reported in [5]. The following notions of *predictable stopping time* and *layer* are instrumental at introducing lattices. There are two major difficulties in extending lattices to Petri nets. Firstly, we lack the notion of time. In automaton unfoldings, time is represented by the length of each considered finite run and therefore is also visible in its finite branching processes as the length of maximal configurations. Unfortunately, no

such facility is available for Petri net unfoldings. Secondly, conditions in Petri net unfoldings are not like conditions in automaton unfoldings: different conditions need to synchronize when chosing an event in their postset. Hence the introduction of lattices will require some effort. In particular, while conditions could be merged in constructing lattices for automata, we need to perform merge at a coarser grain for Petri nets, layers are introduced for this purpose.

Predictable stopping times and layers have been introduced for the first time in [5] to formalize the notion of a "progress of time". In the following, P_c denotes the set of branching places of Petri net $\mathcal{N}$, i.e., places having at least two different transitions in their postsets.

Definition 7 (predictable stopping time) *A branching process $\mathcal{B} = \{B, E, \rightarrow, \varphi\}$ is called a* predictable stopping time *of the unfolding $\mathcal{U}_\mathcal{N}$ if it satisfies the following conditions:*

(i) $\forall b \in B$ *such that* $\varphi(b) \in P_c$, *either* $b^\bullet_\mathcal{B} = \emptyset$ *or* $b^\bullet_\mathcal{B} = b^\bullet$, *where* $b^\bullet_\mathcal{B}$ *denotes the postset of condition b in the branching process $\mathcal{B}$, and $b^\bullet$ denotes, as usual, the postset of condition b in the whole unfolding $\mathcal{U}_\mathcal{N}$.*

(ii) $\mathcal{B}$ is "maximally extended" meaning that each continuation of it contains at least one additional condition labelled by an element of P_c.

The intuition behind the above definition is that branching processes should be regarded "information providers" about the behaviour of the Petri net in consideration. Information is gathered each time a choice is expressed at some branching condition of the unfolding. To gather consistent information, either all alternative choices should be expressed (case $b^\bullet_\mathcal{B} = b^\bullet$ of condition *(i)* of the definition) or no choice should be expressed at all (case $b^\bullet_\mathcal{B} = \emptyset$ of condition *(i)* of the definition). And we should maximally extend the branching process as long as no further information is provided, i.e., until we reach a new condition labelled by a branching place (condition *(ii)* of the definition).

The set of all predictable stopping times is closed under both union and intersection. Having defined our proper notion of time, it is natural to consider the associated notion of "atomic progress of time". We do this next.

Definition 8 (layers) *Let $\mathcal{B}$ and $\mathcal{B}'$ be two predictable stopping times such that 1/ $\mathcal{B}'$ is strictly contained in $\mathcal{B}$, and 2/ there exists no predictable stopping time strictly containing $\mathcal{B}'$ and strictly contained in $\mathcal{B}$. We call a* layer *the following suffix of $\mathcal{B}$:*

$$L \quad = \quad (\mathcal{B} \setminus \mathcal{B}') \cup {}^\bullet(\mathcal{B} \setminus \mathcal{B}') \tag{9}$$

The soundness of this definition relies on the closedness of the set of all predictable stopping times under intersection. The representation (9) of layer L is not unique. However, we note that, if decompositions $L = (\mathcal{B}_1 \setminus \mathcal{B}'_1) \cup {}^\bullet(\mathcal{B}_1 \setminus \mathcal{B}'_1) = (\mathcal{B}_2 \setminus \mathcal{B}'_2) \cup {}^\bullet(\mathcal{B}_2 \setminus \mathcal{B}'_2)$ hold, then L can also be defined using the pair $(\mathcal{B}_1 \cap \mathcal{B}_2, \mathcal{B}'_1 \cap \mathcal{B}'_2)$, hence there exists a *minimal* pair $(\mathcal{B}, \mathcal{B}')$ such that representation (9) of layer L holds, we take it as the *canonical* representation of L and write it

$$L \quad = \quad \mathcal{B}/\mathcal{B}' .$$

The notion of layer is close to, but different from the known notion of *cluster,* see Desel and Esparza [12]. Clusters have been introduced along with free choice Petri nets for analysing blocking. Clusters are obtained as follows : pick a condition and its postset, for each event in this postset, add its corresponding preset, and repeat this process until fixpoint is reached. This is different from our layers, although both notions coincide in many cases. However, our layers are convex, whereas clusters are not guaranteed convex (a subset N of nodes is called convex if $n, n'' \in N$ and $n \preceq n' \preceq n''$ implies $n' \in N$). We collect now some useful results from [5].

- We denote by $\boldsymbol{L}$ the set of all layers of the underlying unfolding. A predictable stopping time is the union of the layers it contains :

$$\mathcal{B} \;=\; \bigcup_{\substack{L \,\in\, \boldsymbol{L} \\ L \,\subset\, \mathcal{B}}} L\,. \tag{10}$$

  Any two layers have disjoint sets of events.

- The set $\boldsymbol{L}$ of all layers is partially ordered as follows : for two layers $L_i = \mathcal{B}_i/\mathcal{B}'_i$, $i = 1, 2$, we set

$$L_1 \preceq L_2 \quad \text{iff} \quad L_1 \subseteq \mathcal{B}'_2\,. \tag{11}$$

  $\boldsymbol{L}$ has a unique minimal element, namely the smallest predictable stopping time. Hence $(\boldsymbol{L}, \preceq)$ is a partial order, we identify it with its transitive reduction which is a directed acyclic graph (DAG). This structuration of time into a DAG once more reflects the partial order nature of time in our theory.

- Layers can be generally infinite, however they are always finite if the following condition is satisfied (in [5], corresponding Petri nets are said to possess a *choice-conformal* unfolding) : for all condition b of unfolding $\mathcal{U}_\mathcal{N}$ such that $\varphi(b) \in P_c$, the restriction, to $b^\bullet$ of the labelling map $e \mapsto \varphi(e)$ is injective, from $b^\bullet$ into $\varphi(b^\bullet)$.

We are now ready to extend lattices to Petri nets. Consider a labelled Petri net $\mathcal{N} = \{P, P_0, T, \to, \lambda\}$ with labelling map taking its values in the alphabet A of alarms, and let $\mathcal{U}_\mathcal{N} = \{B, E, \to, \varphi\}$ be its unfolding. We use the notations of (6). Consider the following equivalence relation on the set $\boldsymbol{L}$ of all layers :

$$\text{for two layers } L_i = \mathcal{B}_i/\mathcal{B}'_i,\ i = 1, 2, \text{ we write } \quad L_1 \quad \sim \quad L_2 \tag{12}$$

if the following two conditions are satisfied — the reader should compare them with the conditions (ℓ_1) and (ℓ_2) at the beginning of this section :

$(\mathbf{L}_1)$ L_1 and L_2 are isomorphic, when seen as occurrence nets (i.e., directed graphs labelled by places and transitions from Petri net $\mathcal{N}$) ; in particular, L_1 and L_2 possess identical continuations ;

($\mathbf{L}_2$) the predictable stopping times $\mathcal{B}'_1$ and $\mathcal{B}'_2$ possess extensions that are alarm-isomorphic (cf. definition 4), i.e., they explain the same alarm pattern.

In the sequel, we denote by $L_{/o}$ the equivalence class of layer L, up to an isomorphism of occurrence nets. The equivalence class of layer L for $\sim$ is denoted by $[L]$. Denote the quotient $\boldsymbol{L}/\sim$ by $\boldsymbol{L}_\sim$, we make it a labelled directed circuitfree *hypergraph* as follows:

(a) If $L \in \boldsymbol{L}$, then

$$[L] \in \boldsymbol{L}_\sim \quad \text{and its label is} \quad \psi([L]) \triangleq L_{/o}\,. \tag{13}$$

(b) For $L \in \boldsymbol{L}$, denote by ${}^\bullet L$ the set of neighbouring ancestors of L in $(\boldsymbol{L}, \prec)$, the set of corresponding equivalence classes for $\sim$ is denoted by $[{}^\bullet L]$. We create a *hyperbranch*

$$[{}^\bullet L] \to [L] \tag{14}$$

and label it with the set

$$\psi([{}^\bullet L] \to [L]) \triangleq \{\, [L' \cap L] \;:\; L' \in {}^\bullet L \,\}\,, \tag{15}$$

where we regard $[L' \cap L]$ as a subnet of $[L']$.

Definition 9 (lattice) *The labelled directed hypergraph $(\boldsymbol{L}_\sim, \to, \psi)$ defined in (13,14,15) is called the* lattice *of the Petri net $\mathcal{N}$, and we denote it by $\mathcal{L}_\mathcal{N}$.*

The unfolding $\mathcal{U}_\mathcal{N}$ can be recovered from $\mathcal{L}_\mathcal{N}$ by using the following recursive algorithm:

Algorithm 1 Assume $\mathcal{L}$ a prefix of $\mathcal{L}_\mathcal{N}$ and $\mathcal{U}$ the corresponding prefix of $\mathcal{U}_\mathcal{N}$.

1. Choose $[L] \in \min(\mathcal{L}_\mathcal{N} \setminus \mathcal{L})$, and get its label $\psi([L])$.
2. Using the inductive construction, for each hyperbranch $[{}^\bullet L] \to [L]$ and associated label $\psi([{}^\bullet L] \to [L])$, construct the corresponding set $C_{[{}^\bullet L] \to [L]}$ of conditions belonging to $\mathcal{U}$.
3. For each hyperbranch $[{}^\bullet L] \to [L]$, continue $\mathcal{U}$ by glueing $\psi([L])$ at $C_{[{}^\bullet L] \to [L]}$.

This yields an extension of $\mathcal{U}$ by the set of layers belonging to equivalence class $[L]$, we denote it by $\mathcal{U} \bullet [L]$, it corresponds to the extension of $\mathcal{L}$ by $[L]$ in $\mathcal{L}_\mathcal{N}$.

Since $\mathcal{L}_\mathcal{N}$ is a coding of the unfolding $\mathcal{U}_\mathcal{N}$, we can use it to code the set of all configurations of $\mathcal{U}_\mathcal{N}$. We are now ready to use lattices for diagnosis. We are given an alarm labelled Petri net $\mathcal{N} = \{P, P_0, T, \to, \lambda\}$, and an associated alarm pattern $\mathcal{A}$. Using theorem 2 we consider the diagnosis net $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, and represent it by its lattice $\mathcal{L}_{\mathcal{N}\times\mathcal{A}}$ instead, this is summarized in the following theorem:

Theorem 3 (diagnosis lattice) *For $\mathcal{N} = \{P, P_0, T, \rightarrow, \lambda\}$ an alarm labelled Petri net, and $\mathcal{A}$ an associated alarm pattern, we have*

$$\bigcup_{\mathcal{A}' \preceq \mathcal{A}} \mathsf{diagnosis}\,(\mathcal{A}') \quad \sim \quad \mathcal{L}_{\mathcal{N} \times \mathcal{A}}\,,$$

where $\mathcal{L}_{\mathcal{N} \times \mathcal{A}}$ is the lattice of product net $\mathcal{N} \times \mathcal{A}$, we call it a diagnosis lattice.

Diagnosis lattices provide an alternative data structure for asynchronous diagnosis. This data structure is linear and not branching any more, it is therefore more effective than unfoldings. For the case of automata they identify with ordinary lattices used in Viterbi-type algorithms for maximum likelihood estimation. Unfortunately their use raises problems for Petri nets having a choice-*non*conformal unfolding, since layers can be infinite in this case, and therefore are not practical any more. Using the preselection technique as in [5] can solve this problem, however.

The notions of predictable stopping time, layer, and lattice, are illustrated in the figures 14, 15, and 16, sitting at the end of the paper. In these figures $(\boldsymbol{L}, \preceq)$ is a tree and $(\boldsymbol{L}_\sim, \rightarrow, \psi)$ is a DAG, not a general hypergraph. The use of diagnosis lattices is illustrated in the figure 17.

6 Algorithms

In this section, we describe in detail the algorithms corresponding to the representations developed in section 4. We first describe the framework we need for the description of these algorithms, it is based on the use of *rewriting rules*.

6.1 Notations

As a prerequisite, we state an inductive construction of the unfolding of a Petri net, as defined in (5) and definition 2 (this construction is borrowed from [16]). In this construction, we make consistent use of the following notations. The conditions of the unfolding have the form (e, p), where e is an event of the unfolding and p a place of the Petri net in consideration; the label of condition (e, p) is p (written $\varphi(e, p) = p$), and its unique input event is e. Conditions $(\perp, p)$ are those having no input event, i.e., the distinguished symbol $\perp$ is used for the minimal conditions of the occurrence net. Similarly, events of the unfolding have the form (X, t), where X is a co-set of conditions belonging to the unfolding, and t is a transition of the Petri net in consideration; the label of event (X, t) is t (written $\varphi(X, t) = t$), and its set of input conditions is X.

In the sequel we make use of these notations and represent a branching process as a pair (B, E) of conditions and events. The set of *branching processes* of $\mathcal{N} = \{P, P_0, T, \rightarrow\}$ can be inductively constructed as follows:

- The following term is a branching process of $\mathcal{N}$:

$$(\{(\perp, p), p \in P_0\}, \emptyset)\ . \tag{16}$$

- If (B,E) is a branching process, $t \in T$ an event of $\mathcal{N}$, and $X \supseteq B$ a co-set labelled by ${}^\bullet t$, then the following term is also a branching process of $\mathcal{N}$:

$$(\, B \cup \{\, (e,p) \,|\, p \in t^\bullet \,\}\,, E \cup \{e\}\,)\,, \quad \text{where} \quad e = (X,t)\,. \tag{17}$$

If $e \notin E$ we call e a *possible extension* of (B,E), written $(B,E) \odot e$, we denote the corresponding extended branching process by $(B,E) \bullet e$ and call it a *continuation* of (B,E) by e. This inductive construction is restated, for convenience, in the following form of a rewriting rule :

$$\frac{\text{precondition}}{\text{current branching process} \;\vdash\; \text{continuation}}$$

Using this notation, rule (17) for branching process continuation rewrites as follows :

$$\frac{X \text{ co-set of } B\,,\ \varphi(X) = {}^\bullet t}{\begin{array}{c} e = (X,t) \ \text{ in :} \\ (B,E) \;\vdash\; (\, B \cup \{\, (e,p) \,|\, p \in t^\bullet \,\}\,, E \cup \{e\}\,) \end{array}} \tag{18}$$

NOTATION. To shorten the rewriting rules, we shall discard the statement "X co-set of B" in the sequel. Be careful that checking such a condition requires knowing the concurrence relation, or, equivalently, the causality and conflict relations.

6.2 Asynchronous diagnosis

We first discuss the

> computation of the unfolding $\mathcal{U}_{\mathcal{N}_1 \times \mathcal{N}_2}$, for two nets $\mathcal{N}_1$ and $\mathcal{N}_2$ having no shared places.

We have two nets $\mathcal{N}_1$ and $\mathcal{N}_2$: p_i (resp. t_i) shall denote generically a place (resp. event) of net i, the labelling map for transitions is denoted by λ, and the labelling map for the unfolding under construction is denoted by φ. Using these notations, we have the following rules for constructing $\mathcal{U}_{\mathcal{N}_1 \times \mathcal{N}_2}$:

$$\frac{\lambda(t_i) \text{ is private, and } \varphi(X_i) = {}^\bullet t_i}{\begin{array}{c} e_i = (X_i, t_i) \ \text{ in :} \\ (B,E) \;\vdash\; (\, B \cup \{\, (e_i,p_i) \,|\, p_i \in t_i^\bullet \,\}\,, E \cup \{e_i\}\,) \end{array}} \tag{19}$$

$$\frac{\lambda(t_1) = \lambda(t_2) \stackrel{\Delta}{=} \lambda\ ,\ \forall i = 1,2\ :\ \varphi(X_i) = {}^\bullet t_i}{\begin{array}{c} e = (X_1 \cup X_2, t)\,,\ \lambda(t) = \lambda \ \text{ in :} \\ (B,E) \;\vdash\; (\, B \cup \{\, (e,p) \,|\, p \in t_1^\bullet \cup t_2^\bullet \,\}\,, E \cup \{e\}\,) \end{array}} \tag{20}$$

Rule (19) performs a local continuation involving a single component, whereas rule (20) performs a synchronized continuation. Thanks to theorem 2,

the above rules (19,20) can be specialized to implement the computation of the unfolding $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$:

$$\frac{\lambda(t)=\lambda(a)\stackrel{\Delta}{=}\alpha\ ,\ \varphi(X)={}^\bullet t\ ,\ \varphi(X^\alpha)={}^\bullet a}{\begin{array}{c}e=(X\uplus X^\alpha,t)\ \text{ in :}\\ (B,E)\ \vdash\ (\,B\cup\{\,(e,p)\,|\,p\in t^\bullet\uplus a^\bullet\,\}\,,E\cup\{e\}\,)\end{array}} \tag{21}$$

We have discarded the rule (19) because $\mathcal{A}$ has no private label, hence rule (19) never applies. This was a first solution to the asynchronous diagnosis problem.

The resulting algorithm can be run "on-line", meaning that the unfolding $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$ can be updated using rule (21) each time a new alarm is collected. Since continuations can occur from any node in the current status of the unfolding, the whole unfolding must be continuously maintained. Of course this data structure is of rapidly increasing complexity, and this makes the general algorithm based on the above rule quite cumbersome. Also, in this general case, explanations of an alarm can occur with an arbitrary long delay, this is the price to pay for accepting unrestricted asynchrony together with no assumption on the sensoring system.

However, for centralized diagnosis, it is a natural assumption that data are sequentially collected at a sensor, in an order which does not contradict the causality relations due to the underlying Petri nets, see the discussion in the beginning of section 4. In the next subsection we shall take advantage of this and provide in this case a much more efficient algorithm.

6.3 Asynchronous diagnosis, an optimized version

More precisely, we investigate the

computation of the diagnosis net $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, for $\mathcal{A}$ a totally ordered alarm pattern of $\mathcal{N}$.

In this subsection we show how to compute the special structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, in the particular case considered. We proceed by starting from the unfolding $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, and we use theorem 2 which relates this unfolding to $\mathcal{U}_{\mathcal{N},\mathcal{A}}$. Also, our objective here is to refine the rules to take advantage of the particular assumptions and derive some optimizations. This is performed in several steps.

1. **On-line computation of $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$.** Consider first rule (21). Since $\mathcal{A}$ is now assumed to be totally ordered, we can simplify the precondition $\varphi(X^\alpha)={}^\bullet a$ and the postcondition. Simply write

$$\begin{array}{rcl}\mathcal{A} & = & (B,E)\,,\ \text{where}\\ B & = & (\bot,1),((1,\alpha_1),2),\ldots,((n,\alpha_n),n),\ldots\\ E & = & (1,\alpha_1),(2,\alpha_2),\ldots,(n,\alpha_n),\ldots,\ \text{and}\ \lambda((n,\alpha_n))=\alpha_n\,,\end{array}$$

and rewrite (21) as follows:

$$\frac{\lambda(t) = \lambda((n, \alpha_n)) = \alpha_n \ , \ \varphi(X) = {}^\bullet t}{\begin{array}{c} e = (X \uplus \{n\}, t) \ \text{ in :} \\ (B, E \ \vdash \ (B \cup \{ (e,p) \,|\, p \in t^\bullet \} \cup \{(e, n+1)\} \, , \, E \cup \{e\}) \end{array}} \tag{22}$$

The presence of index n suggests to consider an *on-line* version of this algorithm. This consists in applying the algorithm in the following way:

$$[\, \forall n = 1, 2, \ldots [\, \forall t \quad : \quad \mathbf{R}_{(22)}(n, t) \,] \,], \tag{23}$$

where $\mathbf{R}_{(22)}(n,t)$ denotes rule (22), for n and t seen as parameters. Then the on-line version is just the inner part of (23):

$$[\, \forall t \quad : \quad \mathbf{R}_{(22)}(\mathit{current}, t) \,], \tag{24}$$

where "*current*" denotes the current value for index n. The continuations performed by applying rule (24) are called the *fresh continuations.*

2. **Computing really** diagnosis$(\mathcal{A})$. Now, reading carefully theorems 1 and 2 reveals that we need in fact to compute $\mathcal{U}_{\mathcal{N}\mathcal{A}}$, not $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$. In $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, some configurations explain only prefixes of alarm pattern $\mathcal{A}$, not the full alarm pattern. These configurations must be removed while computing $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$. To kill the nodes that cannot explain the whole alarm pattern, but only prefixes of it, we only need, after having applied rule $[\forall t : \mathbf{R}_{(22)}(n,t)]$, to discard those nodes that cannot explain the current alarm n: since $\mathcal{A}$ is totally ordered, such nodes will never be able to explain alarm n. Therefore, call E_n the set of events that have been added to E while applying rule $[\forall t : \mathbf{R}_{(22)}(n,t)]$, and let "$\#(E_n)$ in (B,E)" denote the set of nodes, which belong to (B,E) and are in conflict with every node of E_n. We claim the following:

$$\begin{array}{l} \textit{the nodes belonging to "} \ \#(E_n) \ \textit{in} \ (B,E) \textit{"} \\ \textit{will never explain alarm } n. \end{array} \tag{25}$$

Accordingly, the following postprocessing is applied after $[\forall t : \mathbf{R}_{(22)}(n,t)]$:

$$\mathbf{post}\, \mathbf{R}_{(22)}(n) \quad : \quad (B,E) \ \longrightarrow \ (B,E) \setminus (\, \#(E_n) \text{ in } (B,E) \,) \tag{26}$$

It remains to justify claim (25). By definition of E_n, no *immediate* continuation of E_n can explain the nth alarm event. Assume further continuing (B,E) can *later* explain the nth alarm event. This means that there exists a subsequent alarm event, with index $m > n$, and some configuration κ of (B,E), which 1/ continues (B,E), 2/ can explain the nth alarm event, and 3/ such that $e_n \succ e_m$ in κ (with obvious notations). The conjunction of $e_m \succ e_n$ in $\mathcal{A}$, and $e_n \succ e_m$ in κ, contradicts condition 3 of definition 5. This justifies claim (25).

3. **Making use of theorem 1.** Also, a representation equivalent to $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, but having the form of a net, is obtained in the following way. Remove, from $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, the causalities that are inherited from alarm pattern $\mathcal{A}$, and keep only the conflict relations inherited from $\mathcal{A}$. To this end, rule $\mathbf{R}_{(22)}(n,t)$ is rewritten as follows:

$$\mathbf{R}_{(27)}(n,t) \quad : \quad \frac{\lambda(t)=\lambda((n,\alpha_n))=\alpha_n \ , \ \varphi(X)={}^\bullet t}{\begin{array}{c} e=(X\uplus\{n\},t) \ \text{ in :} \\ (B,E) \ \vdash \ (\, B\cup\{\,(e,p)\,|\,p\in t^\bullet\,\} \ , \ E\cup\{e\}\,) \end{array}} \tag{27}$$

Compare with (22): we have removed the causalities from e to index $n+1$, this were the causalities inherited from the totally ordered alarm pattern. On the other hand, the causalities from n to e are kept, this encodes the reinforced conflict relation $\#_{\mathcal{A}}$ introduced in theorem 1. At this stage, rule (24) rewrites:

$$[\,\forall t \,:\, \mathbf{R}_{(27)}(\mathit{current},t)\,] \quad ; \quad \mathbf{post}\,\mathbf{R}_{(27)}(\mathit{current}) \tag{28}$$

where $\mathbf{post}\,\mathbf{R}_{(27)}$ is given in (26).

4. **Optimizing.** We can still optimize rule (28) by noting that, in the term (B,E) resulting from aplying this rule, not all places from B can serve for future continuations of (B,E). Denote by $\perp\!\!\!\perp(E_n)$ the maximal places belonging to (B,E) that are concurrent with *some* event belonging to (E_n) (note the difference with the former definition of $\#(E_n)$). Then we claim that

$$\begin{array}{l} \textit{only the nodes belonging to } E_n^\bullet \cup \perp\!\!\!\perp(E_n) \textit{ in } (B,E), \\ \textit{can serve for future continuations of } (B,E). \end{array} \tag{29}$$

Using claim (29), rule (27) rewrites as follows, note the modification of the precondition:

$$\mathbf{R}^{\mathbf{opt}}_{(27)}(n,t) \quad : \quad \frac{\begin{array}{c} X\subseteq E^\bullet_{n-1}\cup\perp\!\!\!\perp(E_{n-1}) \ \text{ in } (B,E) \\ \lambda(t)=\lambda((n,\alpha_n))=\alpha_n \ , \ \varphi(X)={}^\bullet t \end{array}}{\begin{array}{c} e=(X\uplus\{n\},t) \ \text{ in :} \\ (B,E) \ \vdash \ (\, B\cup\{\,(e,p)\,|\,p\in t^\bullet\,\} \ , \ E\cup\{e\}\,) \end{array}} \tag{30}$$

Then, postprocessing (26) applies after rule (30) as well:

$$[\,\forall t \,:\, \mathbf{R}^{\mathbf{opt}}_{(27)}(\mathit{current},t)\,] \quad ; \quad \mathbf{post}\,\mathbf{R}_{(27)}(\mathit{current}) \tag{31}$$

Now it remain to justify claim (29). To this end, consider the $n+1$st alarm event, and assume there exists some continuation e of (B,E), e explains the $n+1$st alarm event, but it is not the case that e can be concatenated to (B,E) by using the extended precondition (30). Then it must be the case that e has in its prefix some place q strictly anterior to E_n for the causality relation $\preceq$. Hence we must have $e\in\#(E_n)$, and we derive a contradiction as in the proof of claim (25).

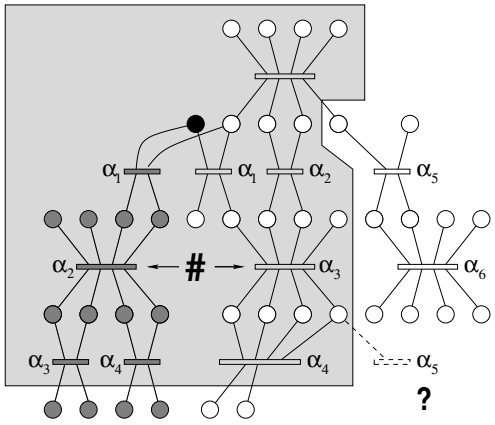

Figure 13: The pruning mechanism.

The pruning mechanism of rule (30) is illustrated in figure 13. In this figure, alarms $\alpha_i, i = 1,2,3,4$ have been processed yet, and next alarm for processing is α_5. No more concatenation can be performed from the light grey zone. In particular, the dashed continuation with a question mark is not possible, as it would give raise to a configuration not explaining α_4.

5. **Maintaining co-sets.** So far we have considered implicit the maintenance of co-sets, see the remark following formula (18) at the beginning of this section. The reason was that the co-set relation can be deduced from the graph structure of the pair (B,E). However since only a postfix of the whole branching process is maintained (post-processing $\mathbf{post}\,\mathbf{R}_{(27)}(n)$ applies), we need in fact to maintain the co-set relation explicitly. We discuss this now, for the case of on-line diagnosis. Our starting point is rule (31). The first rule in (31) refines as follows:

$$\mathbf{R}^{\mathbf{refine}}_{(32)}(n,t) \quad : \quad \frac{\begin{array}{c} X \subseteq E^{\bullet}_{n-1} \cup \perp\!\!\!\perp(E_{n-1}) \ \text{ in } (B,E) \\ \lambda(t) = \lambda((n,\alpha_n)) = \alpha_n \ , \ \varphi(X) = {}^{\bullet}t \end{array}}{\begin{array}{c} e = (X \uplus \{n\}, t) \ \text{ in :} \\ (B,E) \ \vdash \ (\, B \cup \{\, (e,p) \,|\, p \in t^{\bullet} \,\} \ , \ E \cup \{e\}\,) \\ \perp\!\!\!\perp_{\mathcal{A}}(e) \ = \ \bigcap_{b \in X} (\perp\!\!\!\perp_{\mathcal{A}}(b)) \\ \forall b' \in e^{\bullet} \ : \ \perp\!\!\!\perp_{\mathcal{A}}(b') \ = \ \perp\!\!\!\perp_{\mathcal{A}}(e) \bigcup (e^{\bullet} \setminus \{b'\}) \end{array}} \tag{32}$$

where the notation $\perp\!\!\!\perp_{\mathcal{A}}(e)$ denotes the set of nodes of (B,E) which are $\mathcal{A}$-concurrent with e, cf. theorem 1, and the notation X denotes generically an $\mathcal{A}$-co-set. Note that two events explaining the same alarm α_n are not declared $\mathcal{A}$-concurrent, and, since they are not causally related, they are in $\mathcal{A}$-conflict, as requested by definition 6. The refined on-line algorithm is then the following:

$$[\ \forall t \ : \ \mathbf{R}^{\mathbf{refine}}_{(32)}(\mathit{current},t)\ ] \quad ; \quad \mathbf{post}\,\mathbf{R}_{(27)}(\mathit{current},t)\,, \tag{33}$$

and the postprocessing is now fully compatible with the refined rule $\mathbf{R}^{\mathbf{refine}}_{(32)}(n,t)$. This detailed specification supports the use of the special event structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ proposed by theorem 1.

7 Dynamically changing models

For large and complex systems, the model to be used cannot be defined statically, i.e., it cannot be considered to be given once for all. For instance, telecommunications networks are subject to frequent reconfigurations, updates, and upgrades. Therefore it is desirable and sometimes requested that diagnosis techniques can comply with such a situation. This was not the case for our approach so far. Therefore we wish to extend it in order to handle situations in which *the model is finite "locally in time", but may change from time to time.*

We first discuss this in within our original framework of systems and tiles, as introduced in section 2. The idea is that the system $\Sigma = \langle V, X_0, \mathcal{T}\rangle$ itself becomes dynamic. Therefore we modify the definition (1) of tiles as follows. A *dynamic tile* is a 5-tuple

$$\tau \quad = \quad \langle V^-, V\,;\, x^-_{V^-}, \alpha, x_V\rangle \tag{34}$$

where

- $V^-, V \subset \mathcal{V}$ are the finite subsets of *previous* and *current* variables. The important fact is that we can have $V \neq V^-$, meaning that variables can be created or destroyed, dynamically.
- The triple $(x^-_{V^-}, \alpha, x_V)$ is a partial transition, relating the previous V^--state $x^-_{V^-}$ to the current V-state x_V, and emitting event α where α ranges over some set A of possible event labels.

Then the set $\mathcal{T}$ itself is infinite, but each tile τ is obtained from a finite set of prototype tiles by using renaming (i.e., simply changing the names of the variables, not their domains). This model dynamic instanciation of finitely many prototype tiles. We can cast this situation in our Petri net framework, by following mutatis mutandis the reasoning of subsection 2.2. Repeating the reasoning of the latter subsection, we obtain the following class of Petri nets.

Definition 10 (locally finite Petri nets) *Let $\mathcal{N} = \{P, P_0, T, \rightarrow, \lambda\}$ be a Petri net such that: the sets of places P and transitions T are* infinite, *but the initial marking P_0 is* finite, *and each* finite *$X \subset P$ and $S \subset T$ have* finite *pre/postsets ${}^\bullet X/X^\bullet$ and ${}^\bullet S/S^\bullet$, respectively. A Petri net satisfying the above conditions is called* locally finite.

This being said, the analysis of sections 4 and 5, and the algorithms of section 6 extend without modification to locally finite Petri nets and their unfoldings. The following remarks hold:

1. For general algorithms of subsection 6.2, since explanations of alarms can occur with arbitrary long delay, the algorithm still needs to maintain a growing number of candidate transitions.

2. However, for the case of subsection 6.3, where alarms are collected sequentially and an on-line version of the algorithm is applied, only the currently active transitions need to be stored for the algorithm, and this is finite, locally in time, and not growing. The corresponding implementation is therefore adequate to comply with dynamic reconfigurations.

8 Discussion

A net unfolding approach to on-line asynchronous diagnosis was presented. Asynchronous diagnosis was approached by means of hidden state history reconstruction from alarm observations. This true concurrency approach is suitable to distributed systems in which no global state and no global time is available, and therefore a partial order model of time is considered. The work in [17] was a first approach toward solving this problem.

In the present paper, our basic tool was that of net *unfolding* proposed by Mc Millan and Engelfriet, and subsequently developed by Esparza and Römer. Net unfoldings are a branching structure suitable to represent the set of runs of a Petri net using an asynchronous semantics with local states and partial ordered time. Diagnosis nets were introduced as a way to encode all solutions of a diagnosis problem. We have also proposed a new data structure, called diagnosis lattice. This data structure is not branching any more, it is rather linear and properly generalizes the lattice structures used for the on-line diagnosis of automata. Then we have shown that our approach naturally adapts to the case of dynamically changing system model. Finally, as the construction of the system model is a difficult issue per se, we have proposed the use of our tile model as an intermediate step toward building the model. Tiles are partial transitions, and constitute a behavioral information naturally associated to classes of model components in an object oriented modelling approach.

Several questions remain. First of all, this study is clearly an intermediate step toward *distributed* diagnosis, in which diagnosis is perfomed jointly by a network of supervisors communicating asynchronously. Second, the robustness of our algorithms against alarm losses, or more generally the failure to communicate, needs to be investigated. Also, due to the systems complexity, there is little hope indeed, that an exact model can be provided, hence we need to develop diagnosis methods that work based on an incomplete model, i.e., a model not able to explain all observed behaviours. Finally, a probabilistic extension would be welcome. The distributed case together with some of the above extensions are discussed in the report [1].

9 Figures illustrating the different notions

In these figures, time progresses downwards.

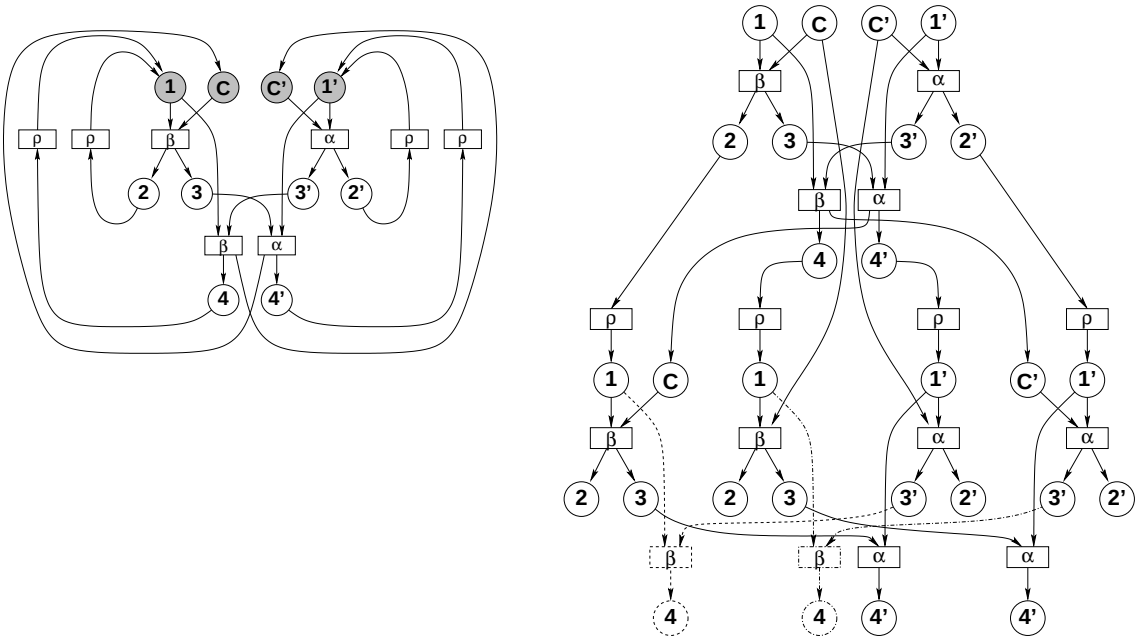

Figure 14: Example 2, and its unfolding. Example 2 (left) is a symmetrized version of example 1, figure 10, in which fault propagation can occur in both directions, through places 3 and 3'. A branching process of example 2 is shown (right). While only places 1,1' are branching in the Petri net, conditions labelled by C,C',3,3' are infinitely branching as well in the unfolding, due to synchronizations. In the unfolding, some places/branches/events are dotted, this has no particular meaning, and is only intended to help for reading.

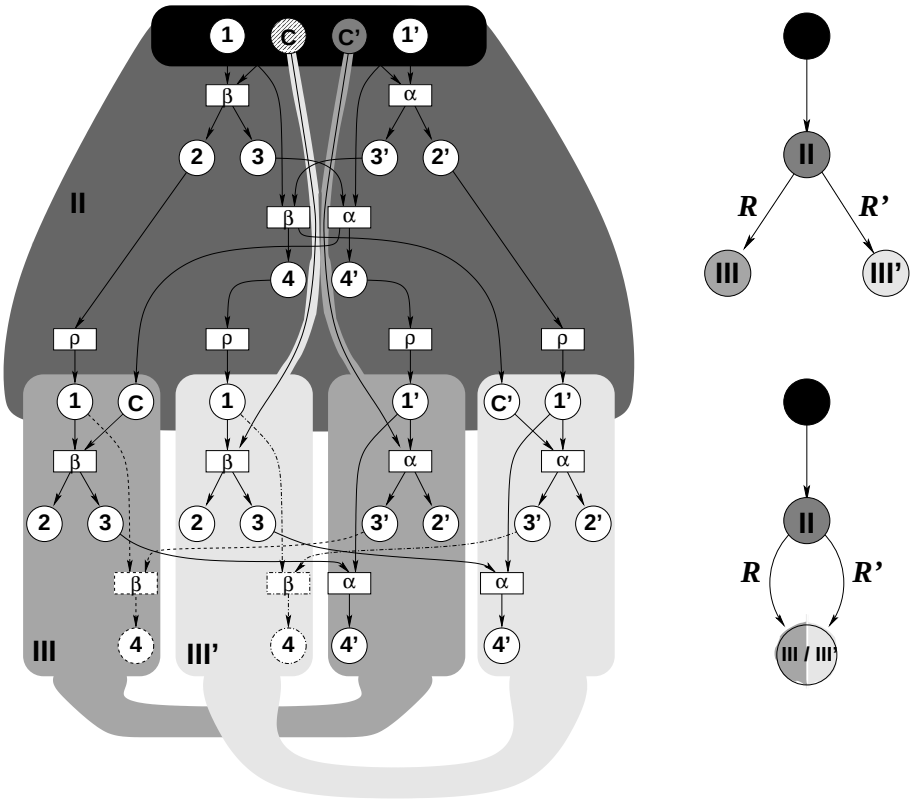

Figure 15: Unfolding of example 2, and lattice. We show again the branching process of example 2 (left). We also show the first four layers, labelled I, II, III, III', respectively, filled with four different colours (note the minimal conditions labelled by C and C', which belong to layers I,II,III and I,II,III', respectively. Accordingly, four stopping times are shown : I, I∪II, I∪II∪III, and I∪II∪III'. Using the partial order defined in (11), these four layers are organized into a tree (top right), where branches are labelled by the roots, ***R,R'***, of the layers. Corresponding details are given in figure 16. Layers III and III' are equivalent according to relation $\sim$ defined in (12). These two layers can be merged for further continuation, and the lattice follows accordingly (bottom right).

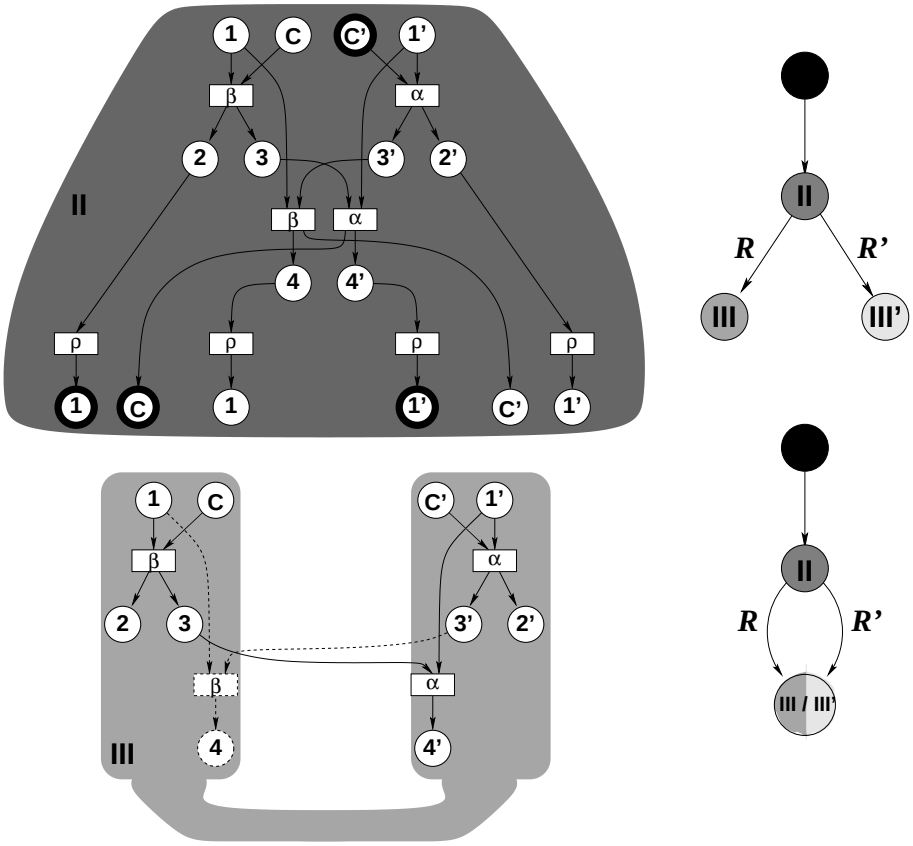

Figure 16: Example 2, layers II, III, and the lattice. We show again the lattice of the figure 15, together with layers II,III, in detail. The *root* $\boldsymbol{R}$ of layer III is composed of the conditions encircled in thick in layer II. Check again that layers III and III' are equivalent according to relation $\sim$ defined in (12).

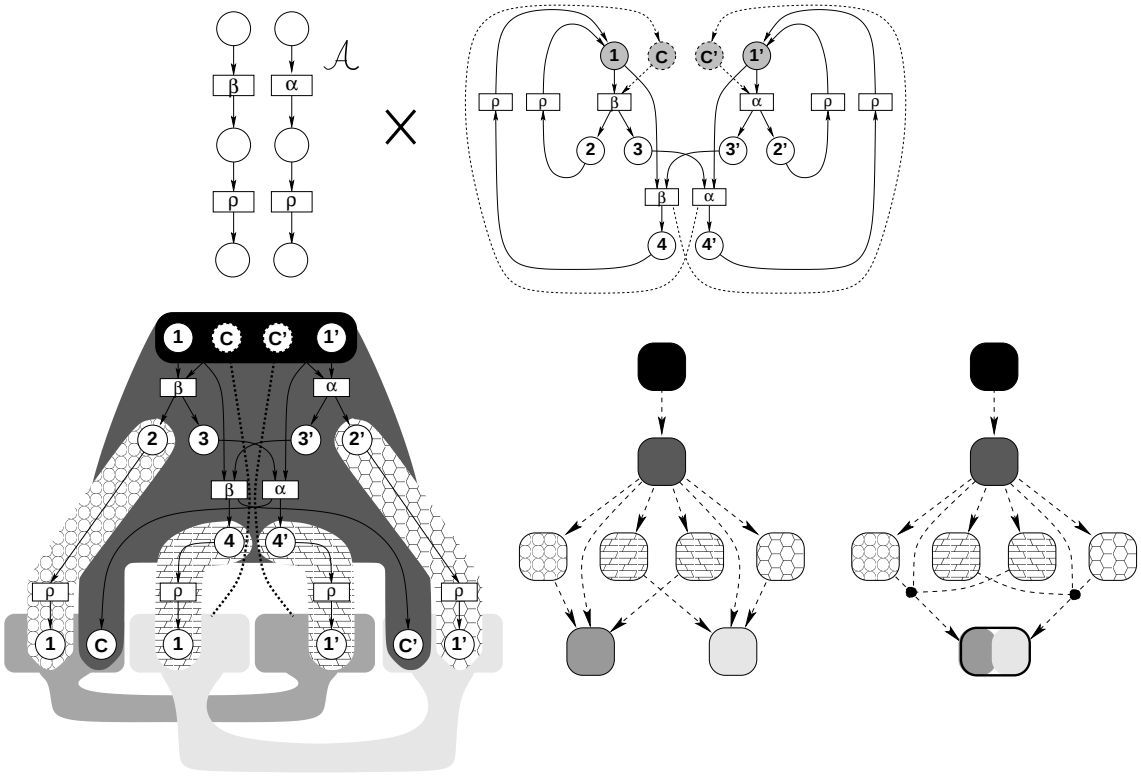

Figure 17: Example 2, diagnosis lattice. We show an alarm pattern for example 2 (top left), it consists of two concurrent alarm sequences $\mathcal{A} = \{\beta \to \rho, \alpha \to \rho\}$, respectively recorded by two local sensors. Then, we show again the net $\mathcal{N}$, we need to compute the product $\mathcal{A} \times \mathcal{N}$, and then its unfolding. The result is too intricate, and therefore we show instead (bottom left) the subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$. In reading this net, one should keep the following in mind : each event labelled by ρ can explain several alarms, and therefore it should be duplicated (we do not show this). Then the different layers are superimposed on $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$, shown by different grey styles. Let us focus on the layer covering the path $2 \to \rho \to 1$, it represents in fact two copies of this path branching at condition labelled by 2, corresponding to the matching with the two different alarms labelled by ρ, hence we have a layer branching at condition labelled by 2. A similar explanation holds for the other layers. Compare with the layers of figure 15, and note that layers are now smaller, due to additional branching originating from the matching with the alarms labelled by ρ. Then we show the DAG $(\boldsymbol{L}, \preceq)$ (bottom middle), the last two layers can be merged. Accordingly, we show the diagnosis lattice, i.e. the hypergraph $\boldsymbol{L}_{\sim}$ (bottom right). For the sake of clarity, we do not show the labels associated with the hyperbranches.

References

[1] E. Fabre, A. Benveniste, C. Jard, and M. Smith. "Diagnosis of distributed discrete event systems, a net unfolding approach". Preprint, Feb. 2001.
`http://www.irisa.fr/sigma2/benveniste/pub/F&al2001.html`

[2] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, C. Jard. A Petri net approach to fault detection and diagnosis in distributed systems. Part II : extending Viterbi algorithm and HMM techniques to Petri nets. *CDC'97 Proceedings,* San Diego, Dec. 1997.

[3] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, C. Jard. Fault Detection and Diagnosis in Distributed Systems : an Approach by Partially Stochastic Petri nets, *Discrete Event Dynamic Systems: theory and application,* special issue on Hybrid Systems, vol. 8, pp. 203-231, June 98.

[4] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence* 110: 135-183, 1999.

[5] A. Benveniste, E. Fabre, and S. Haar. "Markov nets : probabilistic models for distributed and concurrent systems". Preprint, Feb. 2001.
`http://www.irisa.fr/sigma2/benveniste/pub/BlGFH2000.html`

[6] R. Boubour, C. Jard, A. Aghasaryan, E. Fabre, A. Benveniste. A Petri net approach to fault detection and diagnosis in distributed systems. Part I : application to telecommunication networks, motivations and modeling. *CDC'97 Proceedings,* San Diego, Dec. 1997.

[7] A.T. Bouloutas, G. Hart, and M. Schwartz. Two extensions of the Viterbi algorithm. *IEEE Trans. on Information Theory,* 37(2):430–436, March 1991.

[8] A.T. Bouloutas, S. Calo, and A. Finkel. Alarm correlation and fault identification in communication networks. *IEEE Trans. on Communications,* 42(2/3/4), 1994.

[9] C. Cassandras and S. Lafortune. *Introduction to discrete event systems.* Kluwer Academic Publishers, 1999.

[10] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems: theory and application.* 10(1/2), 33-86, 2000.

[11] R. Debouk, S. Lafortune, and D. Teneketzis. On the effect of communication delays in failure diagnosis of decentralized discrete event systems. Control group report CGR00-04, Univ. of Michigan at Ann Arbor, submitted for publication, 2001.

[12] J. Desel, and J. Esparza. *Free Choice Petri Nets.* Cambridge University Press, 1995.

[13] R.G. Gardner, and D. Harle. Methods and systems for alarm correlation. In *GlobeCom 96,* London, November 1996.

[14] J. Engelfriet. *Branching Processes of Petri Nets.* Acta Informatica 28, 1991, pp 575–591.

[15] J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. In T. Margaria and B. Steffen Eds., *Proc. of TACACS'96,* LNCS 1055, 87-106, 1996. Extended version to appear in *Formal Methods in System Design,* 2000.

[16] J. Esparza, and S. Römer. An unfolding algorithm for synchronous products of transition systems, in *proceedings of CONCUR'99,* LNCS 1664, Springer Verlag, 1999.

[17] E. Fabre, A. Benveniste, C. Jard, L. Ricker, and M. Smith. Distributed state reconstruction for discrete event systems. Proc. of the *2000 IEEE Control and Decision Conference (CDC'2000),* Sydney, Dec. 2000.

[18] K.X. He and M.D. Lemmon. Liveness verification of discrete-event systems modeled by n-safe Petri nets. in *Proc. of the 21st Int. Conf. on Application and Theory of Petri Nets,* Danmark, June 2000.

[19] K.X. He and M.D. Lemmon. On the existence of liveness-enforcing supervisory policies of discrete-event systems modeled by n-safe Petri nets. in *Proc. of IFAC'2000 Conf. on Control Systems Design,* special session on Petri nets, Slovakia, June 2000.

[20] I. Katsela, A.T. Bouloutas, and S. Calo. Centralized vs distributed fault localisation. *Integrated Network Management IV,* A.S. Sethi, Y. Raynaud, and F. Faure-Vincent, Eds. Chapman and Hall, 251-261, 1995.

[21] K. McMillan. *Symbolic Model Cheking: an approach to the state explosion problem.* Kluwer, 1993.

[22] Y. Pencolé. Decentralized diagnoser approach: application to telecommunication network. In Proceedings of the International Workshop on Principles of Diagnosis (DX'00), Morelia, Mexico, 2000.

[23] W. Reisig. *Petri nets.* Springer Verlag, 1985.

[24] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. Autom. Control* 40(9), 1555-1575, 1995.

[25] R. Sengupta. Diagnosis and communications in distributed systems. In *Proc. of* WODES *1998, international Workshop On Discrete Event Systems,* 144-151, IEE, London, England, 1998.

[26] S. Tripakis. Decentralized observability (and control) of regular languages is undecidable. Univ. of California at Berkeley, preprint, saubmitted for publication, 2001.

[27] G. Winskel. Event structures. In *Advances in Petri nets,* LNCS vol. 255, 325–392, Springer Verlag, 1987.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399